# MULTI SENSOR FUSION BASED FRAMEWORK FOR EFFICIENT MOBILE ROBOT COLLISION AVOIDANCE AND PATH FOLLOWING SYSTEM

Marwah Almasri

Under the Supervision of Prof. Khaled Elleithy

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

November, 2016

MULTI SENSOR FUSION BASED FRAMEWORK FOR EFFICIENT MOBILE ROBOT COLLISION AVOIDANCE AND PATH FOLLOWING SYSTEM

Marwah Almasri

Under the Supervision of Prof. Khaled Elleithy

## Approvals

### Committee Members

| Name | Signature | Date |
|---|---|---|
| Prof. Khaled Elleithy | | 11/21/16 |
| Prof. Junling Hu | | 11/18/2016 |
| Prof. Xingguo Xiong | | 11/18/2016 |
| Prof. Miad Faezipour | | 11/18/2016 |
| Prof. Magdy Bayoumi | | 11/30/16 |

### Ph.D. Program Coordinator

Dr. Khaled M. Elleithy 12/6/16

### Chairman, Computer Science and Engineering Department

Dr. Ausif Mahmood 12-8-2016

### Dean, School of Engineering

Dr. Tarek M. Sobh

# MULTI SENSOR FUSION BASED FRAMEWORK FOR EFFICIENT MOBILE ROBOT COLLISION AVOIDANCE AND PATH FOLLOWING SYSTEM

MULTI SENSOR FUSION BASED FRAMEWORK FOR EFFICIENT MOBILE ROBOT COLLISION AVOIDANCE AND PATH FOLLOWING SYSTEM

# ABSTRACT

The field of autonomous mobile robotics has recently gained the interests of many researchers. Due to the specific needs required by various applications of mobile robot systems (especially in navigation), designing a real-time obstacle avoidance and path following robot system has become the backbone of controlling robots in unknown environments. Therefore, an efficient collision avoidance and path following methodology is needed to develop an intelligent and effective autonomous mobile robot system. Mobile robots are equipped with various types of sensors (such as GPS, camera, infrared and ultrasonic sensors); these sensors are used to observe the surrounding environment. However, these sensors sometimes fail and have inaccurate readings. Therefore, the integration of sensor fusion will help to solve this dilemma and enhance the overall performance.

A new technique for line following and collision avoidance in the mobile robotic systems is introduced. The proposed technique relies on the use of infrared sensors and

involves a reasonable level of calculations, to be easily used in real-time control applications.

In addition, a fusion model based on fuzzy logic is proposed. Eight distance sensors and a range finder camera are used for the collision avoidance approach, where three ground sensors are used for the line or path following approach. The fuzzy system is composed of nine inputs (which are the eight distance sensors and the camera), two outputs (which are the left and right velocities of the mobile robot's wheels), and twenty four fuzzy rules for the robot's movement.

Webots Pro simulator is used for modeling the environment and robot to show the ability of the robot to follow a path, detect obstacles, and navigate around them to avoid collision. It also shows that the robot has been successfully following extremely congested curves and has avoided any obstacle that emerged on its path.

The proposed methodology which includes the collision avoidance based on fuzzy logic fusion model and line following robot, has been implemented and tested through simulation and real-time experiments. Various scenarios have been presented with static and dynamic obstacles, using one and multiple robots while avoiding obstacles in different shapes and sizes. The proposed methodology reduced the traveled distance of the mobile robot, as well as minimized the energy consumption and the distance between the robot and the obstacle detected as compared to a non-fuzzy logic approach.

# DEDICATION

***To my mother (Efaqat Ezmirly), and to the memory of my father (Mohammad),***

***For all of your constant love and support.***

# ACKNOWLEDGEMENTS

My thanks are wholly devoted to God who has helped me all the way to complete this work successfully.

I would like to express my special appreciation and thanks to my advisor Prof. Khaled Elleithy for his continuous support, patience, and motivation towards my Ph.D study and research. I am indebted to him for sharing his expertise and valuable guidance. Without his support and insight, this work would have never been accomplished. Prof. Khaled Elleithy, I would like to thank you very much for the dedicated time, assistance, and thoughtful comments you provided throughout the process.

I would also like to extend my appreciation to Prof. Magdy Bayoumi for his support, feedback, and insightful comments about my work. I am expressing my deepest gratitude to the committee members: Prof. Junling Hu, Prof. Xingguo Xiong, and Prof. Miad Faezipour; for their guidance, suggestions, and support.

A special thanks goes to my lovely mother, brothers (Ahmad, Bakr, and Omar), and sisters (Maha, Manal, and Rawan) for their support over the past years. Words cannot express how grateful I am for your prayers and encouragement; that is what sustained me this far. Finally, I would like to thank my best friend and sister Abrar Alajlan for the constant support and encouragement. I am sure that our friendship will last a lifetime.

# ACRONYMS

| | |
|---|---|
| DoS | Denial of Service |
| ATR | Automatic Target Recognition |
| MAP | Maximum A Posteriori |
| ML | Maximum Likelihood |
| pdf | Probability density function |
| SMC | Sequential Monte Carlo |
| MMSE | Minimum Mean Square Error |
| EKF | Extended Kalman Filter |
| UKF | Unscented Kalman Filter |
| MLE | Maximum Likelihood Estimator |
| DSP | Digital Signal Processing |
| DSC | Distributed Source Coding |
| DISCUS | Distributed Source Coding Using Syndromes |
| FTI | Fault-Tolerant Interval Function |

| | |
|---|---|
| Webots Pro | Mobile robot simulator software |
| GUI | Graphical User Interface |
| LFA | Line Follower Approach |
| CCA | Collision Avoidance Approach |
| TIME_STEP | The time of one step of the simulation time |
| d | Number of distance sensors |
| g | Number of ground sensors |
| thr | Collision avoidance threshold |
| s | Mobile robot’s global speed |
| fs | Mobile robot’s offset speed |
| ls | Mobile robot’s left motor speed |
| rs | Mobile robot’s right motor speed |
| Δ | The difference between the right and left ground sensors |
| ANFIS | Adaptive Neuro-Fuzzy Inference System |
| AGV | Autonomous Ground Vehicle |
| TAV | Teleoperated Autonomous Vehicle |

| | |
|---|---|
| OAM | Obstacle Avoidance Module |
| LFM | Line Flowing Module |
| LEM | Line Entering Module |
| LLM | Line Leaving Module |
| UTM | U-Turn Module |
| FLS | Fuzzy Logic System |
| LV | Left velocity |
| RV | Right velocity |
| FIS | Fuzzy Inference System |
| OBSNF | Obstacle not found membership function |
| OBSF | Obstacle found membership function |
| Near | Range finder camera near distance membership function |
| FAR | Range finder camera far distance membership function |
| NEG_V | Negative velocity |
| POS_V | Positive velocity |

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1: INTRODUCTION

There has been a spurt of interest in recent years in the area of autonomous mobile robots that are considered as mechanical devices capable of completing scheduled tasks, decision-making and navigating without any involvement from humans [1], [2]; This has brought up some serious concerns about the interaction between mobile robots and the environment [3]. Due to the increased demand of this type of robot, various techniques and algorithms were developed. Most of them are focused on navigating the robot in collision-free trajectories, with the controlling of the robot's speed and direction [4].

In contrast to regular robots, an autonomous robot does not know the surrounding environment in advance; hence, it needs to be programmed in a way that it can adjust and be flexible to all changes that occur during operation [5]. Mobile robot motion planning in an unknown environment has always been the main research focus in the mobile robotics area, due to its practical importance and the complex nature of the problem. Several collision avoidance and line following techniques have been introduced lately.

Each of these techniques was developed to be used in specific applications for the purposes of education, entertainment, business, etc [3].

The ability to detect obstacles in real-time mobile robotics systems is a very critical requirement for any practical application of autonomous vehicles. The main objective behind using the obstacle avoidance approach is to obtain a collision-free trajectory from the starting point to the target in monitoring environments. There are two types of obstacles: static obstacle, which has a fixed position and requires a priori knowledge of the obstacle; dynamic obstacle, which does not require any priori knowledge of the motion of the obstacle and has uncertain motion and patterns (moving objects). Indeed, detecting dynamic obstacles is more challenging than detecting static obstacles; the dynamic obstacle has a changeable direction and requires a prediction of the obstacle position at every time step in order to achieve the requirement of a time-critical trajectory planning [6].

In addition, path planning in mobile robots can be divided into two types based on the robot's knowledge of the environment. Global path planning is where the environmental information is predefined; global path planning is also called static collision avoidance planning. Local path planning is where the environmental information is not pre-known and is also known as dynamic collision avoidance planning. Local path planning is more demanding than global path planning. Local path planning quickly takes into consideration some kinds of measurements regarding the dimensions of the moving obstacle (such as position, size and shape) through sensors to avoid unknown obstacles while the robot is moving towards the goal state [6].

Various methods have been proposed as solutions for line following problems, generally including line-following, object-following, and path tracking, etc. The line follower in robotics is simply an autonomous mobile robot that detects a particular line and keeps following it. This line can be as visible as a black line in a white area or as invisible as a magnetic field [3].

Line following in mobile robots can be achieved in three basic operations. First, capture the line width by camera (image processing) or some reflective sensors mounted at the front of the robot. Second, adjust the robot to follow the predefined line by using some infrared sensors placed around the robot. Third, control the robot speed based on the line condition [3].

### 1.1 Research Problem and Scope

The development of autonomous mobile robots is still at the center of numerous research projects to provide a collision free path. Given the specific needs required by different applications of mobile robots (especially in navigation), it is crucial to develop an autonomous robotic system that is capable of avoiding obstacles while following a path in real-time applications. Consequently, an efficient collision avoidance and path following technique is essential to assure intelligent and effective autonomous mobile robot systems [3].

These robots need to communicate with the environment through various sensor modules. In the area of autonomous mobile robotics, the most significant tasks are collision avoidance, path planning, and obstacle detection. They autonomously allow the

robot to avoid or mitigate a collision, while traveling in an efficient path towards a desired goal. The robot can be mounted by different kinds of sensors in order to observe the surrounding environment, thus steering the robot accordingly. However, many factors affect the reliability and efficiency of these sensors. The integration of a multi-sensor fusion system can overcome this problem by combining inputs coming from different types of sensors, hence having more reliable and complete outputs; this plays a key role in building a more efficient autonomous mobile robotic system [4].

### 1.2 Motivation behind the Research

To assure efficiency and robustness, the integration of sensor or data fusion is considered a crucial aspect (especially in real-time systems). Sensor fusion can be defined as the combining process of sensory data in order to generate improved data that assures robustness and confidence, as well as diminishes ambiguity and uncertainty. For the purpose of collision avoidance and path following approaches, different types of sensors (such as camera, infrared sensor, ultrasonic sensor, and GPS) can detect different aspects of the environment. Each sensor has its own capability and accuracy, whereas integrating multiple sensors enhances the overall performance and detection of obstacles [4].

Many data fusion techniques exist such as fuzzy logic, kalman filter, particle filter, Bayesian methods, and the Dempster-Shafer methods. Using one of these techniques or a combination is useful for enhancing the output data. Fuzzy logic is an approach used for sensor fusion with small computational load, especially in unknown environments to avoid faulty interventions resulting from invalid observations. Therefore,

using fuzzy logic to fuse data obtained from different types of sensors boosts the robustness of the algorithm required for collision avoidance and path planning in autonomous mobile robotic systems.

### 1.3 Contributions of the Proposed Research

The goal of this work is to design and experimentally implement a collision-free path follower robot, with the integration of data fusion. The aim of the fusion system is to enhance the robustness and the efficiency of the algorithm used.

A novel real-time obstacle avoidance and line follower approach for mobile robots has been developed and tested in both simulation and real-time experiments. The proposed technique allows the simultaneous detection of obstacles with the control of the mobile robot to eliminate collisions and recover the path again. The novelty of this technique arises from the integration of data fusion techniques, along with the proposed algorithm for steering the mobile robot. This combination is extremely helpful for obtaining more accurate sensor data, thus enabling the robot to react more efficiently in case of obstacle detection. The performance of the proposed technique has been evaluated using simulated and experimental data. Our framework aims for higher efficiency and accuracy using data fusion, while ensuring overcoming obstacles along the path. In this work, we are mainly concerned about extracting necessary fused sensor data from multi-modality sensors for vigorous collision free trajectory planning approach.

The contribution of the proposed technique is as follows:

- Design a collision avoidance and line follower framework for a mobile robot.
- The capability of the mobile robot to avoid obstacles along its path, based on the use of infrared sensors.
- The capability of the mobile robot to follow a path.
- An efficient algorithm of the proposed technique is developed.
- The integration of multisensory information from range finder camera and infrared sensors, using fuzzy logic fusion system for collision avoidance and line follower mobile robot.
- The proposed methodology develops membership functions for inputs and outputs and designs fuzzy rules, based on these inputs and outputs.
- The performance of the mobile robot when programmed with the fuzzy logic sets and rules.
- The proposed methodology has been successfully tested in Webots Pro simulator and real-time experiments.
- The proposed methodology has been tested in different levels of complexities with static and dynamic obstacles, using one and multiple robots while avoiding obstacles in different shapes and sizes.
- The proposed methodology reduced the traveled distance of the mobile robot, as well as minimized the energy consumption and the distance between the robot and the obstacle detected as compared to a non-fuzzy logic approach.

# CHAPTER 2: LITERATURE SURVEY

## 2.1 Introduction

Robotic systems need a lot of information about the surrounding environment in order to be able to accomplish the required tasks (such as collision avoidance, line following, or target seeking). This information must be acquired faster, more accurately, and more precisely. Therefore, heterogeneous sensors or many homogenous sensors are used to get useful measurements about the environment. However, data can be redundant and overlapped, which consumes more energy and time. To eliminate this problem, sensor fusion is applied to combine data from different types of sensors to generate more complete data.

Sensors are used to observe the surrounding environment. Sometimes sensors fail to collect accurate data from the environment, due to pressure and temperature. In other cases, this failure can be attributed to electromagnetic noise or radiation. Therefore, all readings and measurement would be inaccurate and inefficient. In order to overcome these problems, data fusion (which is a technique combining data from several sources to be more accurate and complete) is used. Data fusion is applied in centralized systems, as well as in distributed systems [7]. Data fusion can eliminate redundant data and thus save energy, which results in an improved performance [8].

Data fusion has been used in many detection applications such as robotics [9]. Recently, new applications such as Denial of Service (DoS) detection deployed the data fusion concept successfully [10]. Another example is intrusion detection [11].

In relation to the importance of data fusion, this section presents many techniques that have been applied in sensor based systems in general. Our goal is to analyze each technique and evaluate the advantages and the disadvantages of each in order to comprehend the best usability of these techniques in different applications [12].

## 2.2 Data Fusion Techniques and Methods

Based on the purpose of the method, data fusion techniques can be implemented for a variety of "objectives such as inference, estimation, classification, feature maps, abstract sensors, aggregation, and compression" [13]. In this section, many techniques used in data fusion are discussed along with their applications. Figure 2.1, shows all data fusion techniques.

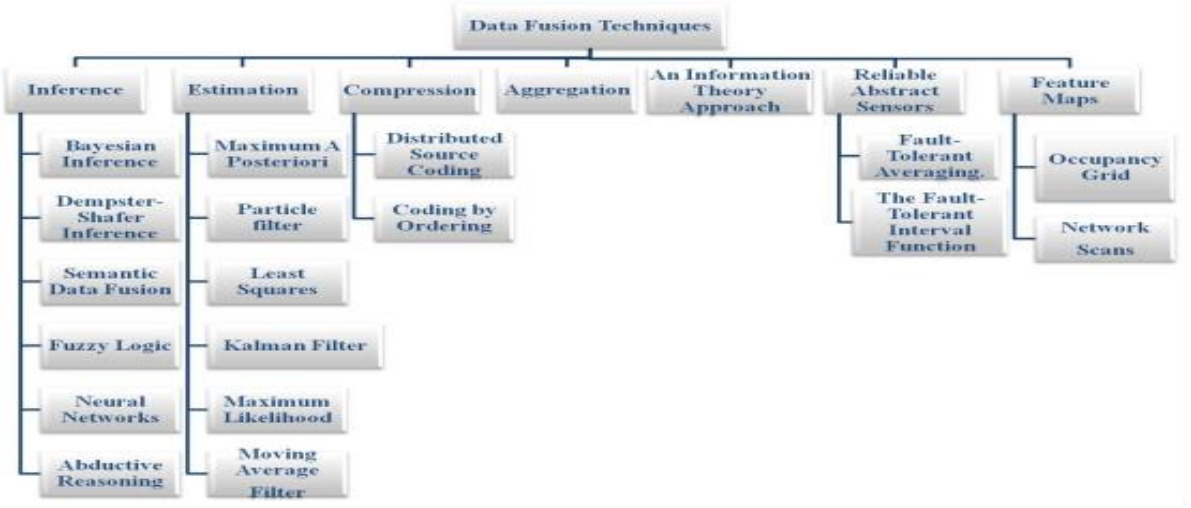



Figure 2.1. Data fusion techniques.

### 2.2.1 Inference Methods

Inference method is mostly used in decision fusion where a decision is generated depending on the perceived situational knowledge. "Classical inference methods are based on Bayesian inference and Dempster-Shafer Belief Accumulation theory" [13],[14]. Other inference methods such as fuzzy logic, neural networks, abductive reasoning, and semantic data fusion are also highlighted.

#### 2.2.1.1 Bayesian Inference

Depending on the probability theory, Bayesian Inference merges all evidences where the uncertainty in Bayesian Inference describes the belief. It assumes the value of 0 for absolute disbelief and 1 for absolute belief. Bayesian inference is basically based on the "Bayes' rule" [15], [13], which is represented in Equation (2.1):

$$Pr(B \mid A ) = (Pr(A \mid B ) * Pr(B )) / ( Pr(A)) \quad (2.1)$$

Where, Pr(B | A ) is the belief of hypothesis B given the information A, Pr(A | B ) is the probability of receiving A, given that B is true, Pr(B ) is the prior probability, and Pr(A) is the normalizing constant.

The critical issue in Bayesian Inference is that the probabilities Pr (A) and Pr (A|B) should be estimated because they are unknown. The neural network approach has been used to guess the conditional probabilities for the decision-making process in Bayesian inference module [16]. In addition, Cou´E et al. [17] used Bayesian programming in fusing data from various sensors such as laser and video in order to obtain more reliable and accurate data.

#### 2.2.1.2 Dempster-Shafer Inference

This method is based on the "Dempster-Shafer Belief", which generalizes the Bayesian theory. Dempster-Shafer Belief was proposed by both Dempster [18] and Shafer [19]. Dempster-Shafer Inference introduces a formalism that is applied for incomplete knowledge and evidence combination [20]. An important factor in Dempster-Shafer method is the set of all possible states which further demonstrates the system. This set is called the ‘frame of discernment’. The elements of the power set of possible states are called hypotheses. Each hypothesis has its assigned probability. In addition, the belief function which is called ‘bel’ is defined by Dempster-Shafer and also the degree of doubt ‘dou’ that is based on the belief function are [21].

#### 2.2.1.3 Semantic Data Fusion

Semantic data fusion is done as an in-network inference. The semantic data fusion method is composed of two important phases. The first phase is called knowledge base construction, which collects the "knowledge abstractions" into a form of semantic data. The second phase is called pattern matching (inference), which uses the semantic data provided by the previous phase to fuse relevant attributes for pattern matching [22]. This method was first introduced by Friedlander and Phoha [22] for target classification. Friedlander [23] explains many techniques that extract semantic data from sensors by converting sensor data into formal languages. He applies these techniques for the recognition of the robots’ behavior and for saving resources.

#### 2.2.1.4 Fuzzy Logic

Fuzzy logic deals with "approximate reasoning" in order to obtain "conclusions from imprecise premises" [24], [7]. Zadeh [25] has introduced the concept of fuzzy sets which later guided him to the fuzzy logic theory. The data fusion algorithm based on fuzzy logic theory has four main phases: "fuzzification", "rule evaluation", "combination" or "aggregation of rules", and "deffuzification" [26]. In the second phase which is the rule evaluation, the implications or rules are used to process the fuzzified inputs. These rules are in the form of “if A then B”, where A is a conditional statement. Sometimes more than two conditional statements are used which is called complex implications. When applying complex implications, fuzzy operators are used for computing the final result [27]. The most common fuzzy logic inference operators used are shown in Equations (2.2), (2.3), (2.4), (2.5), (2.6), (2.7), (2.8), and (2.9) as follows [27]:

$$x \longrightarrow y = yx \tag{2.2}$$

$$x \longrightarrow y = \min\{1, 1-x+y\} \tag{2.3}$$

$$x \longrightarrow y = \min\{x, y\} \tag{2.4}$$

$$x \longrightarrow y = \begin{cases} 1 & if\ x \leq y \\ 0 & otherwise \end{cases} \tag{2.5}$$

$$x \longrightarrow y = \begin{cases} 1 & if\ x \leq y \\ y & otherwise \end{cases} \tag{2.6}$$

$$x \longrightarrow y = \begin{cases} 1 & if\ x \leq y \\ y/x & otherwise \end{cases} \tag{2.7}$$

$$x \longrightarrow y = \max\{1-x, y\} \tag{2.8}$$

$$x \longrightarrow y = 1\text{-}x+xy \quad (2.9)$$

In Equation (2.4), the Mamdani inference operator is presented. It finds the minimum degree of the membership (x, y). Both Mamdani and Tsukamoto-Sugeno inference methods are based on fuzzy logic [28]. However, the Mamdani method is considered a better method since it ensures an efficient data fusion, extends the sensor lifetime, and reduces delay compared to Tsukamoto method.

In [29], authors use fuzzy logic control and an intelligent sensor network for autonomous navigational robotic vehicle which has the ability of avoiding obstacles. Another implementation of fuzzy logic is for efficient routing that minimizes energy usage [30].

#### 2.2.1.5 Neural Networks

The Neural network is applied in "learning systems" with fuzzy logic to manage its "learning rate" [31], [32], [7]. In the data fusion domain, neural networks have been applied in many applications such as "Automatic Target Recognition (ATR)" [33]. Lewis and Powers [34] fused audio-visual information using neural networks for audio-visual speech recognition.

#### 2.2.1.6 Abductive Reasoning

Abductive Reasoning is the best hypothesis for explaining observed evidence [35]. Figure 2.2 shows the deduction and abduction example. The abductive inference finds the maximum a posteriori probability [36]. Abduction was used in machine learning problems [37] and diagnosis problems [38].

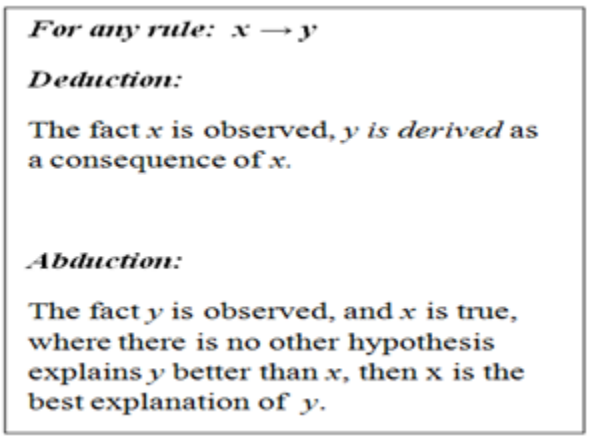



Figure 2.2. The deduction and abduction example.

### 2.2.2 Estimation Methods

Estimation methods are derived from the control and the probability theories in order to calculate a process vector from a series of measurement vectors [39]. Examples of Estimation methods are Maximum A Posteriori (MAP), Particle filter, Least Squares, Kalman filter, Maximum Likelihood (ML), and Moving Average filter. The details of each method are presented in this section.

#### 2.2.2.1 Maximum A Posteriori (MAP)

This technique is based on Bayesian theory. Given that 'a', is the state to estimate, where 'b'= {b(1),b(2),..,b(n)} is a set of n observations of 'a', the MAP estimator is used to figure out a value of 'a' in order to maximize the posterior distribution function [40] as in Equation (2.10).

$$\hat{X}(n)=\arg\max_a \; pdf(a|b) \tag{2.10}$$

Where pdf is the probability density function.

MAP estimator was used by Schmitt et al. [41] in a known environment to locate the joint positions of mobile robots. Another implementation of MAP estimator was by Yuan and Kam [42] in the collision resolution algorithm. The algorithm's purpose is to control the traffic between the fusion node and the source, where MAP estimator figures out the number of nodes that are being transmitted. Therefore, the retransmission probability of these nodes needs to be updated accordingly.

#### 2.2.2.2 Particle Filter

Particle filters are recursive processes of the "sequential Monte Carlo methods (SMC)" [43]. They are suitable for applications that implement a non-Gaussian noise [44]. They use a large number of random measurements which are composed of particles (samples) that are driven from distributions and weights of the particles. The random measurements are helpful in calculating all kinds of unknown estimates such as minimum mean square error (MMSE) and maximum a posteriori (MAP). The Particle filter technique represents significant densities by particles and weights. It then computes the integrals by Monte Carlo methods. There are three important operations of the Particle filters: sample step that generates particles, importance step that computes the particle weights which are normalized later, and the resampling step. The resampling is important as it eliminates the trajectories with small weights and highlights the ones that are dominating [45].

#### 2.2.2.3 Least Squares

The "Least Squares method is a mathematical optimization technique that searches for a function that best fits a set of input measurements. This is accomplished by

minimizing the sum of the square error between points generated by the function and the input measurements" [7]. Unlike the "Maximum A Posteriori Probability", the Least Square does not use any previous probability. Therefore, it works in a deterministic manner [13]. The Least Squares method tries to find the value of x [40] as in Equation (2.11).

$$\hat{x}(n) = \underset{x}{\operatorname{argmin}} \sum_{i=1}^{n} [a(i) - h(i,x)]^2 \tag{2.11}$$

Where h is the sensor model for a sequence of $1 \leq i \leq n$ observations.

An advantage of using the Least Squares method is reducing the communication between the source node and the sink. This is achieved by sharing the sensor data through the linear regression instead of transmitting the actual data [46].

#### 2.2.2.4 Kalman Filter

The Kalman filter was invented by Kalman [47] and it gained popularity as a technique used for data fusion. The Kalman filter is shown in Figure 2.3. Based on some measurement y(n) which is shown in Equation (2.12), and the system parameters (which are known in advance), the estimate of x(n), and the prediction of x(n + 1) are presented in Equations (2.13), and (2.14) respectively.

$$y(n) = H(n)\, x(n) + r(n) \tag{2.12}$$

Where: H(n) is the measurement matrix, r is a random variable that follows the zero-mean Gaussian laws.

$$\hat{X}(n)= \hat{X}(n \mid n-1)+K(n)[y(n)-H(n \hat{X}(n \mid n-1)] \quad (2.13)$$

Where K is the Kalman filter gain.

$$\hat{X}(n+1 \mid n) = Ts(n)\,\hat{X}(t \mid t) + Ti(n)\,I(n) \quad (2.14)$$

Where: Ts(n) is the state transition matrix, Ti (n) is the input transition matrix, and I (n) is the input vector.

The Kalman filter technique works well in a linear model where it retrieves optimal estimates recursively [48]. On the other hand, in a nonlinear model, other methods should be used such as "Extended Kalman filter (EKF)" [49], and the "Unscented Kalman Filter (UKF)" [50].

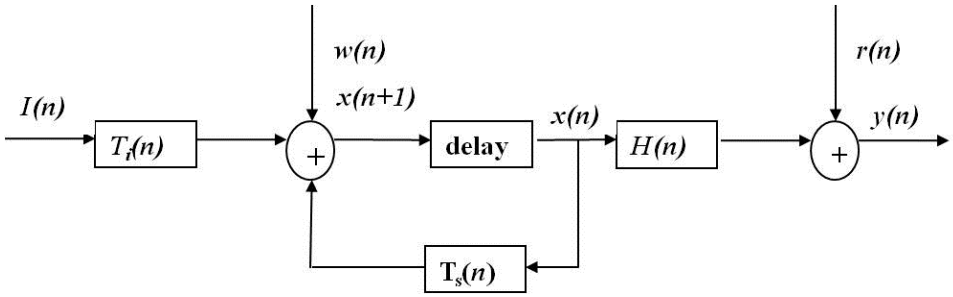



Figure 2.3. Kalman filter block diagram

#### 2.2.2.5 Maximum Likelihood (ML)

To estimate a state ‘a’ as an example, where ‘b’= {b(1),b(2),..,b(n)} is a set of n observations of ‘a’, the likelihood function is defined as follows:

$$\lambda(a) = p(b \mid a) \quad (2.15)$$

Where p is the probability density function.

The Maximum Likelihood estimator (MLE) is used to figure out a value of 'a' in order to maximize the likelihood function [40] as in Equation (2.16).

$$\hat{a}\,(n) = \mathrm{arg}max_{,}\; p(b|a). \tag{2.16}$$

A new distributed and localized MLE was proposed by Xiao et al. [51] with more robustness, where each node can compute a "local unbiased estimate" to eventually reach "the global Maximum Likelihood solution" [13]. This method was further developed by Xiao et al. [52] in order to deliver measurements in a timely manner.

#### 2.2.2.6 Moving Average Filter

The moving average filter is mainly used in "digital signal processing (DSP) solutions" [13]. It has many advantages such that it is easy to use as it reduces "random white noise" while maintaining a "sharp step response" [13]. For this reasons it is an optimal filter in the time domain for processing encoded signals [53]. The true signal x = ( (1), (2), . .) is estimated by Equation (2.17).

$$\hat{x}\,(n) = 1/w \sum_{i=0}^{w-1} z(n-i) \tag{2.17}$$

Where z=(z(1), z(2), . . .), is the input digital signal, w is the filter's window that indicates the number of input observations for every n ≥ w.

In addition, w refers to the number of steps needed for the filter to identify the signal level's variance. As the value of w increases, the signal becomes cleaner. In contrast, as the value of w decreases, the step edge becomes sharper. The Moving Average filter is able to decline √w of the white noise variance [53].

### 2.2.3 Compression

Compression methods are applied through spatially correlating all sensors with no additional communication cost. This can be obtained by providing two sensors with correlated observations [54]. Several compression methods are discussed in this section.

#### 2.2.3.1 Distributed Source Coding (DSC)

Distributed Source Coding (DSC) [55], is "the compression of multiple correlated sources, physically separated, that do not communicate with each other "[56]. One of the most popular data compression methods is the "Distributed Source Coding Using Syndromes" (DISCUS) framework [57]. In DISCUS, assuming we have a sensor X which wants to transmit its observation to sensor Y. In order to code X's observation, X can send only an index. There is one requirement which is the Hamming distance between X and Y which is at most one. This means that the difference of X and Y can be only one bit. Suppose that a sensor observation can be any value of the set S={000, 001, 010, 011, 100, 101, 110, 111}. X and Y have four cosets {000, 111}, {001, 110}, {010, 101}, {100, 011}. As shown in Figure 2.4, sensor X sends the index of 10 which corresponds to the coset of {010, 101}. Y now can decode the index along with its own observation of (100). Since the Hammimg distance should be at most one between the two, Y knows that the value provided by X should be 101 [13].

#### 2.2.3.2 Coding by Ordering

This technique was first introduced in Petrovic et al. [58]. In this technique, each node sends the data to the border node. The border nodes are responsible for sending what is called a supper-packet, which is a group of all packets, to the sink node. Table

2.1, gives an example of coding by order. As shown in Table 2.1, we have four nodes that each of them provides an observation of the value from 0 to 5: X,Y,Z, and W. As shown in Table 2.1, the border node can suppress all values by W. The ordering is 3! which means that we have 6 possible orderings of the three remaining nodes: X, Y, and Z. For example, if the observation value for node W is 1, the packet order is {X,Z,Y} where it can be {Z,X,Y} if the observation value for node W is 4 and so on [13].

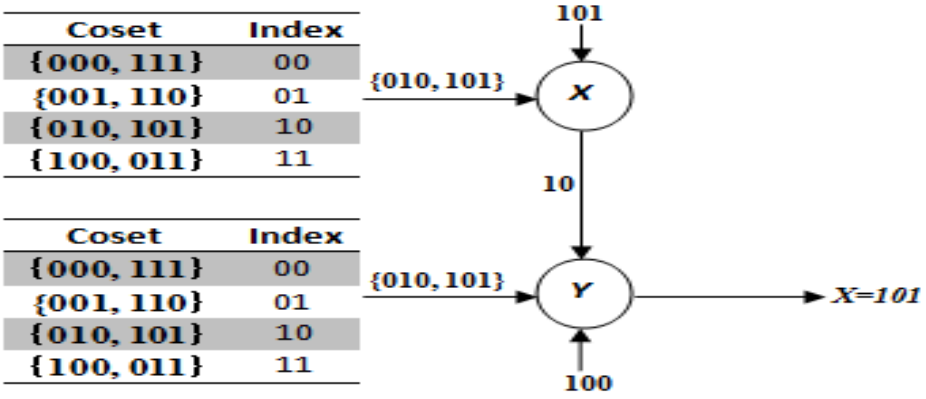



Figure 2.4. An example of DISCUS data compression.

Table 2.1: Code by ordering example.

| Packet Ordering | Observation Value (W) |
|---|---|
| {X,Y,Z} | 0 |
| {X,Z,Y} | 1 |
| {Y,X,Z} | 2 |
| {Y,Z,X} | 3 |
| {Z,X,Y} | 4 |
| {Z,Y,X} | 5 |

### 2.2.4 Aggregation

According to Kulik et al. [59], data aggregations is defined as a technique that is used for solving two kinds of problems: implosion which occurs when the data sensed are duplicated by the same node because of the strategy used in routing, and overlap which occurs when two different nodes broadcast the same data (redundant sensors) [13]. Redundancy has a negative effect as it wastes the energy. Therefore, data aggregation and data fusion are important to reduce energy consumption. For that specific reason, data aggregation is applied for the purpose of reducing redundancy in neighboring nodes [60], [61]. Using data fusion techniques can decrease the number of packets needed to be transmitted by processing data locally and then send only a digest to the sink node which in return saves energy and bandwidth. To illustrate this, the centralized approach takes O (n3/2) bit-hops, where when applying data fusion techniques it takes only O (n) bit–hops for data transmission [62].

In-network data aggregation algorithms have gained a lot of attention recently since they require coordination among nodes when they are distributed in the network to assure high performance which is basically a complex functionality. In-network aggregation can be defined as collecting and routing data within a "multi-hop network" where it processes data at intermediate nodes in order to decrease energy consumption and thus increase the network’s lifetime [63].

### 2.2.5 An Information Theory Approach

Using multiple sensors instead of a single sensor in any network can enhance data and observation reliability. Information fusion based on multiple sensors is harder to estimate in advance. This leads to a probabilistic data collection and processing which can be measured and analyzed by applying the information theory principles [64]. Both the "Information" and "Detection" theories help in solving many problems regarding data fusion. Ahmed and Pottie [65] have used a Bayesian technique for fusion which uses different sensor types along with different sensing capabilities.

### 2.2.6. Reliable Abstract Sensors

This method was first proposed by Marzullo [66] which suggests three different types of sensors: "concrete sensor", which senses the environment by collecting samples of a physical variable, "abstract sensor" which represents the observation in a set of values depending on the concrete sensor, and "reliable abstract sensor" which contains the real values of the physical variable. This type of sensor is computed using a number of abstract sensors. This fusion method has been applied in various applications in time synchronization [67]. Many algorithms and functions are used with reliable abstract sensors for time synchronization such as "Fault-Tolerant Averaging" algorithm and "Fault-Tolerant Interval" (FTI) function.

#### 2.2.6.1 Fault-Tolerant Averaging

This algorithm is used in data fusion methods as it fuses a n number of "abstract sensors" into correct "reliable abstract sensors" even if there are incorrect sensors [66].

The algorithm works as follows. Suppose we have L={I1, . . . , $I_n$} where $I_i = [x_i , y_i]$ by *n* abstract sensors at the same time and we have at most f of *n* abstract sensors which are incorrect or faulty. The "Fault-Tolerant Averaging" algorithm is shown in Equation (2.18) which has a complexity of *O(nlog n)* [66].

$$\mathcal{M}_n^f\ (L) = \{Low, High\} \tag{2.18}$$

Where:

*Low* refers to the smallest value in at least $n - f$ intervals in L, and *High* refers to the largest value in at least $n - f$ intervals in L.

Figure 2.5, shows two different scenarios of applying the Fault-Tolerant Averaging algorithm where there is one faulty sensor. In Figure 2.5 (a) Sen 2 and Sen 3 do not have any intersection; therefore, one of them is the faulty sensor. $\mathcal{M}_4^1$ (sen 1,sen 2,sen 3 ,sen 4) has {*Low*,*High*}, where *Low* (the left edge of Sen 1)= n − f = 4 − 1 = 3, and *High* (the right edge of Sen 4)= n − f = 4 − 1 = 3. However, in Figure 2.5 (b), the right edge of Sen 2 has moved to the left and becomes Sen 2'.

As a result, we have now $\mathcal{M}_4^1$ (sen1,sen 2',sen 3,sen 4) which indicates the instability of *M*. Consequently, the left edge of the result is the left edge of Sen 3 (Low value) and the right edge of the result is the right edge of Sen 4 (High value). This algorithm was further extended by Chew and Marzullo [68] where they fuse data from multidimensional sensors.

#### 2.2.6.2 Fault-Tolerant Interval Function

This function was introduced by Schmid and Schossmaier [69]. The Fault-Tolerant Interval (FTI) function is also used in data fusion methods. Again, we have at most f of *n* abstract sensors considered as incorrect or faulty sensors. FTI function is shown in Equation (2.19).

$\mathcal{F}_n^f$ (L)={*Low*,*High*} (2.19)

Where: *Low* refers to the $(f + 1)^{th}$ largest of the left edges {x1, . . . , xn}

*High* refers to the $(f + 1)^{th}$ smallest of the right edges { y1, . . . , yn}

FTI function indicates that when there are few alterations in the input intervals, unlike the Fault-Tolerant Averaging algorithm, the result will include only few changes as well. As a result, the FTI function is more robust as compared to the Fault-Tolerant Averaging algorithm [69].

Figure 2.6 shows the same example as Figure 2.5; however the result is not that affected when Sen 2' is moved (Figure 2.6(b)). Therefore, FTI obviously is less vulnerable to small alterations in the input intervals as compared to the Fault-Tolerant Averaging algorithm [69].

### 2.2.7 Feature Maps

Sometimes using raw sensory data are not sufficient especially in guidance and resource management applications. As a result, some features that well describe the environment need to be extracted [14]. Many data fusion methods of inference and

estimation produce a feature map. There are two feature maps which are occupancy grid and network scans.

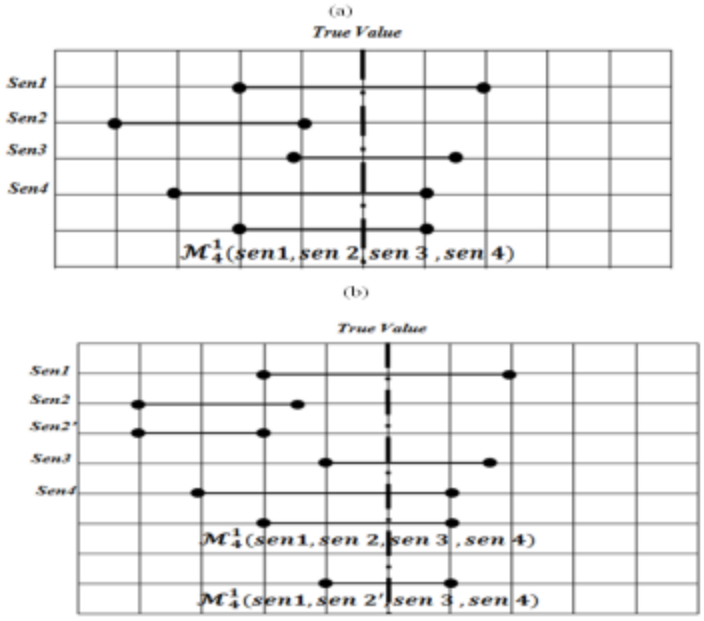



Figure 2.5. Two different scenarios of applying the "Fault-Tolerant Averaging" algorithm where there is one faulty sensor.

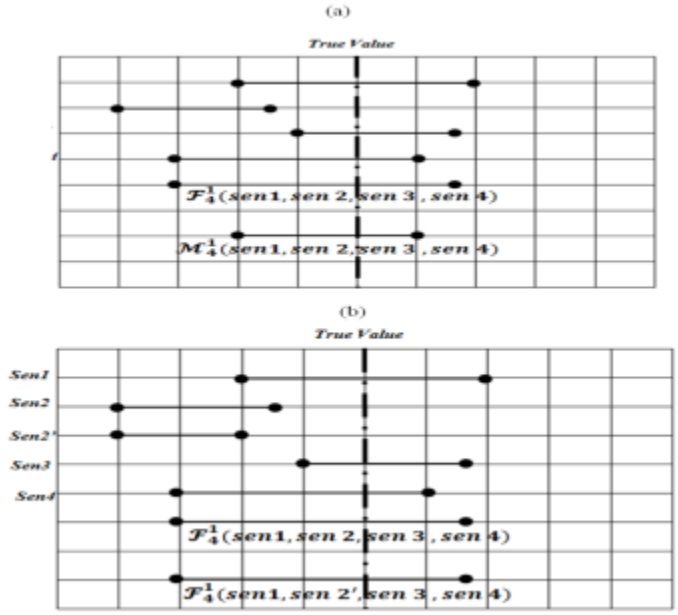



Figure 2.6. Two different scenarios of applying The "Fault-Tolerant Interval" (FTI) function.

#### 2.2.7.1 Occupancy Grid

Occupancy maps define a 2D/3D representation of the space which is organized in square cells where every cell has an estimate that indicates its probabilistic occupancy [70]. This probability is calculated by using multiple types of sensors and various data fusion techniques [71]. Occupancy maps are used in many applications such as robot perception [72], the location's estimation [73], and navigation [74].

#### 2.2.7.2 Network Scans

Network Scans are kinds of activity maps. They also give an overview of the resource distribution in the network [75]. One of the most popular network scans is called eScan [75] which provides information about the remaining energy in the network. The algorithm forms an aggregation tree where each node calculates its local eScan and then sends it to the sink. If two or more eScans are received at the same node, an aggregation process is involved to identify the remaining energy of nodes in a specific region. Finally a map is generated [75].

## 2.3 Evaluation and Comparison of Data Fusion Techniques

This section evaluates all the data fusion techniques and draws a conclusion about which technique is most suitable and reliable to be applied.

Both the "Bayesian Inference" and the "Dempster-Shafer" theories are well-known Inference methods. Dempster-Shafer method generalizes Bayesian Inference. However, "Dempster-Shafer theory" is a more flexible method than "Bayesian Inference" due to its capability to fuse data from various types of sensors unlike Bayesian Inference

[63]. Another difference between these two techniques is that Dempster-Shafer theory does not require assigning apriori probabilities to unknown propositions [14]. In contrast, Dempster-Shafer involves longer calculations [76]. In addition, Fuzzy logic method is best suitable for decision making with uncertain information from multiple sensors. It also improves the quality of information [27]. On the other hand, fuzzy logic cannot solve problems without the knowledge of an expert as it does not have the learning membership function either during solving the problem or after the problem has been solved [77].

Applying neural network has many advantages. In neural network, data fusion is done closely to the source node which results in enhancing its performance. The algorithm used in neural network draws the important features of data and can be adjusted to meet the requirement of various applications [78]. It also provides robustness to handle many issues like noise [79]. It identifies various signals and reduces the errors and false alarm rate of the sensors in an efficient manner [80]. However, many issues need to be considered during the implementation of a neural network such as the problem of local extremum, misclassification due to data dimension increase, and convergence speed of the training [81]. Abductive Reasoning is another technique which works for pattern reasoning more than a data fusion method. It has been used successfully in fault diagnosis and event detection [13]. The semantic data fusion technique has the ability to improve resource utilization especially when collecting and processing data [13]. This method also reduces transmission cost because the nodes transmit formal language structure without the need of transmitting raw data. On the other hand, this technique

requires in some scenarios a known set of behaviors in advance, which is a difficult process in specific situations [82].

Moreover, when the state that needs to be estimated is not based on some random variables, the Maximum Likelihood (ML) technique is suitable to be applied. It also finds the value of this state and assumes it is fixed. In contrast, the "Maximum A Posteriori" (MAP) technique does not consider that the state's value is fixed. On the other hand, it takes it as the result of some random variables with known prior pdf [40]. In addition, the "Least Squares" technique is more accurate and suitable to be applied where the state is fixed. This technique does not use any previous probability as compared to the Maximum A Posteriori (MAP) technique [13]. The Moving Average Filter technique can be used to decrease the random white noise. It has also been used to reduce the errors caused by tracking applications [83]. The downside of this technique is that an old value will have the same impact as the most recent measurement which will affect the final result [84]. Kalman filter is an important and powerful technique as it can estimate past, present, and future states [49]. The Kalman filter can be unstable due to the "critical value for the arrival rate of the observations" [85].

Furthermore, Particle filter is an excellent technique used to overcome some difficult problems such as signal processing, navigation, communications, and computer vision. On the other hand, it has some drawbacks as it is considered a complex technique that has a computational intensity [45].

In addition, even though Occupancy grids show only a restricted class of maps which indicate incorrect independence assumptions in prior and posterior distributions,

they also have the advantage of being simply applied [86]. The network scan technique can be helpful in describing network resources and activity. In particular, eScan can guide designers where to deploy new sensors since it presents low energy regions [75]. Moreover, the Fault-Tolerant Averaging technique can successfully fuse n number of abstract sensors into correct reliable abstract sensors where in fact there are incorrect original sensors [66]. However, few alterations in the input intervals can affect the performance of the "Fault-Tolerant Averaging" algorithm [66]. On the other hand, the Fault-Tolerant Interval Function is more robust due to the fact that few alterations in the input intervals will lead to only few alterations in the output [69]. The aggregation technique helps to eliminate redundancy and traffic load which saves energy. However, by using this technique, the fusion node can be compromised by malicious attackers which affect the correctness of the fusion data. Another disadvantage of this technique is that there might be multiple copies of the same fusion results at the sink node which increases the energy consumption at the sink node [87]. Distributed Source Coding (DSC) has the advantage of making the coding decisions process works efficiently separated from the routing process. On the other hand, it requires more computational complexity. It also needs to collect some data from joint statistics which is not an easy task [63]. The Code by Ordering technique is simple but does not present all possible correlations between sensors [13]. Finally, the information theory approach is suitable for analyzing many problems regarding data collection and processing by multiple sensors [64].

Table 2.2, summaries the advantages and the disadvantages of all data fusion techniques [12]. Based on previous findings, we evaluate the various data fusion techniques discussed in this section and draw a closure [12]. To conclude, there are various data fusion techniques that have been applied. However, some of these techniques do not concern the specific requirements of wireless sensors in robotic applications such as low energy consumption and flexibility. Therefore, for the best applicability of data fusion using wireless sensors in robotic applications, some techniques outweigh others as follows [12]:

- The Dempster-Shafer is a good technique as it fuses data sensed by different types of sensors which are needed in many applications.
- The fuzzy logic technique performs very well in the decision making process and has better data quality.
- Neural networks enhance the process of data fusion which is an advantage as it saves power consumption.
- The Semantic data fusion technique saves resources.
- The Least Squares technique has high accuracy.
- The Moving Average Filter technique decreases the chances of errors which also saves a lot of energy and thus increases the performance.
- The Network scan (eScan) can show low power regions in order to fill in with new full energy sensors.

- The aggregation technique eliminates redundant data and thus saves energy.

Table 2.2. Comparison of data fusion techniques.

| Data Fusion Technique | **Advantages** | **Disadvantages** |
|---|---|---|
| Bayesian Inference | • More accurate than Dempster-Shafer technique | • Does not fuse data from various types of sensors<br>• Needs to assign apriori probabilities to unknown propositions |
| Dempster-Shafer | • Generalizes Bayesian Inference technique<br>• Flexible technique because it has the ability to fuse data from various types of sensors<br>• Does not assign apriori probabilities to unknown propositions | • Less accurate technique as compared to Bayesian Inference<br>• Longer calculations involved |
| Fuzzy Logic | • Effective data fusion technique due to its ability of enhancing the data quality. | • Needs the knowledge of an expert to solve the problem<br>• Learning the membership function is difficult during or after solving the problem |
| Neural Network | • Enhance the performance of data fusion because it is done closely to the source node<br>• The neural network's algorithm is adjustable to the application requirements.<br>• Efficiently decreases the errors and false alarm rate of the sensors | • Many issues need to be solved such as local extremum, misclassification, and convergence speed of the training. |
| Abductive Reasoning | • Successfully used in fault diagnosis and event detection | |
| Semantic Data Fusion | • Improves resource utilization<br>• Reduces transmission cost | • Requires a known set of behaviors in advance, which is a difficult process in specific situations. |
| Maximum Likelihood (ML) | • Suitable when the state is not a random variable<br>• Does not require the sharing of all data | |
| Maximum A Posteriori (MAP) | • The state's value is the result of some random variables with known prior pdf | |
| Least Squares | • Does not use any prior probability as compared to the Maximum A Posteriori (MAP) technique. | |
| Moving Average Filter | • Decreases the random white noise<br>• Reduces the errors caused by tracking applications. | • The final result can be easily affected as the old value will have the same impact as the most recent measurement. |

| | | |
|---|---|---|
| Kalman Filter | • Estimates past, present, and future states. | • It needs clock synchronization which can impact its performance<br>• Unstable due to the critical value found for the arrival rate of the observations |
| Particle Filter | • Can solve some difficult problems such as signal processing, navigation, communications, and computer vision. | • A complex technique that has a computational intensity |
| Occupancy Grids | • Can be simplybe applied | • Shows only a restricted class of maps which presents incorrect independence assumptions. |
| Network Scan | • Describes the network resources and activity.<br>• eScan can guide designers as to where to deploy new sensors as it demonstrates low energy regions | • If two or more eScans are received at the same node, an aggregation process is required in order to determine the remaining energy of the nodes. |
| Fault-Tolerant Averaging | • Fuses several abstract sensors into correct reliable abstract sensors where in fact these abstract sensors are incorrect original sensors. | • The performance can be affected by few alterations in the input intervals |
| Fault-Tolerant Interval | • More robust than the Fault-Tolerant Averaging technique because few alterations in the input intervals will result in few alterations in the output | |
| Aggregation | • Eliminates redundancy and traffic load<br>• Saves energy. | • The fusion node can be compromised by malicious attackers which affect the correctness of the fusion data. |
| Distributed Source Coding (DSC) | • making the coding decisions process works efficiently separated from the routing process | • Requires more computational complexity.<br>• Collects some data from joint statistics which is not an easy task |
| Code by ordering | • Simple technique | • Does not present all possible correlations between sensor nodes |
| Information Theory Approach | • Analyzes problems in data collection and processing by multiple sensors. | |

# CHAPTER 3: ROBOTIC PLATFORM

Deploying autonomous mobile robots is coupled with the use of external sensors that assist in detecting obstacles in advance. The mobile robot uses these sensors to receive information about the tested area through digital image processing or distance measurements to recognize any possible obstacles [88]. Several ways of testing the surroundings have been introduced in the literature of path planning of mobile robots. Although ultrasonic sensors, positioning systems, and cameras are most widely used to move in an unknown environment, they are not the most suitable solution to facilitate the robot. Therefore, some infrared sensors are used to follow an optimal non-collision path from source to destination, according to particular performance objectives [89].

The most well-known sensors used to follow a specific path while detecting obstacles and measuring the distance between robots and objects are infrared, ultrasonic, and laser sensors [3].

Obstacles detected can be moving or static objects in known or unknown environments. In addition, the path planning behavior can be categorized as global path planning (where the environment is entirely known in advance), or local path planning

(where the environment is partly known or not known at all). The later case is called dynamic collision avoidance [90].

### 3.1 Robot and Environment Modeling

Using simulations to test the proposed technique is very useful, prior to investigations with real robots. They are more convenient to use, less expensive, and easier to setup. In this work, Webots Pro simulator is used to develop a line follower and collision avoidance environment. It is one of the most well-known simulation software used in mobile robots that is developed by Cyberbotics [91].

It is a Graphical User Interface (GUI), which creates an environment that is suitable for mobile robot simulation. It also allows the creation of obstacles in different shapes and sizes. The mobile robot used in Webots Pro simulator is called the E-puck robot, which is equipped with a large choice of sensors and actuators (such as camera, infrared sensors, GPS, and LED sensors) [91].

The environment is modeled with a white floor that has a black line, in order for the robot to follow it. It also has solid obstacles, which the robot should avoid them. The environment in Webots Pro is called “world”; a world file can be built using a new project directory. Each project file is composed of four main windows (as shown in Figure 3.1): the Scene tree represents a hierarchical view of the world, the 3D window demonstrates the 3D simulation, the Text editor has the source code (Controller), and the Console shows outputs and compilation [91].

The distance sensors used for collision avoidance are 8 infrared sensors placed around the robot. Figure 3.2 shows a semantic drawing of the top view of the robot used in this work, which is the “E-puck” robot. The red lines represent the directions of infrared distance sensors. For simplicity purposes, we grouped all sensors based on their positions [3].

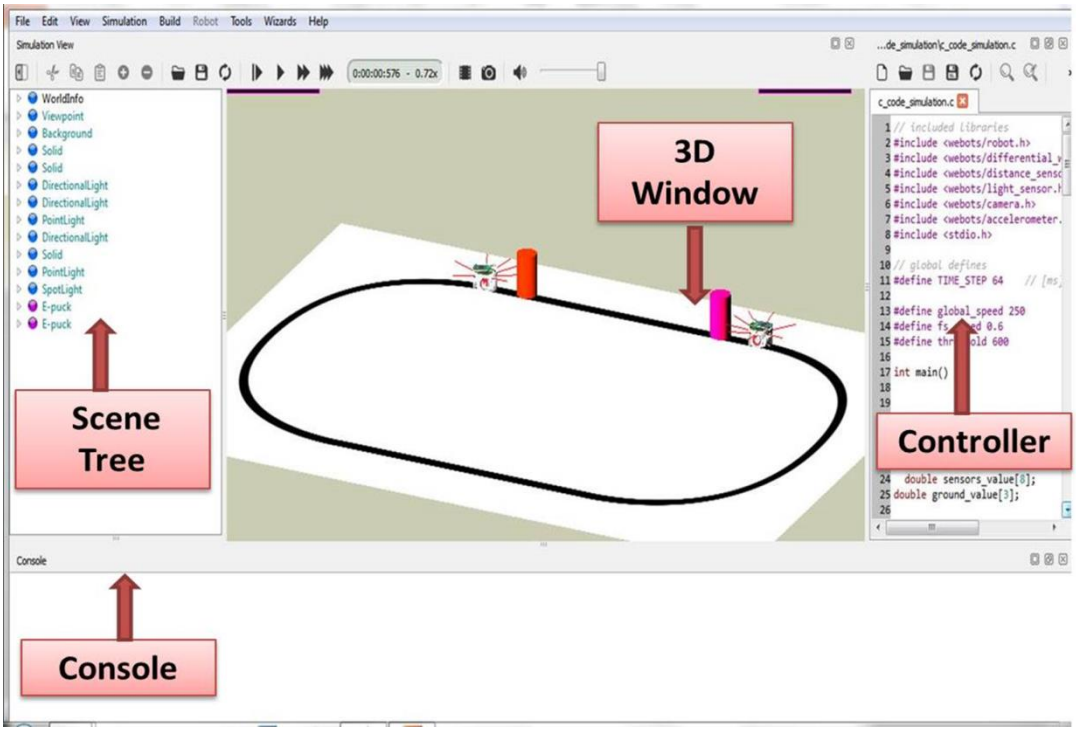



Figure 3.1. Webots Pro simulator overview

Moreover, the robot detects the obstacles based on the values returned by these sensors, that ranged between 0 and 2000; the values returned from distance sensors depend on the distance between the robot and the obstacle. In other words, the values returned will be 0 ( no light) if there is no obstacle detected and 1000 means obstacle is too close to the robot (big amount of light is measured). For the line-following approach, another type of infrared sensor called ground sensors is used. They are three proximity

sensors mounted on the frontal area of the robot and pointed to the ground, in order to detect the line as shown in Figure 3.3. These sensors allow the robot to see the color level of the ground at three locations in the line across its front [3]. Figure 3.4 shows the real E-puck robot.

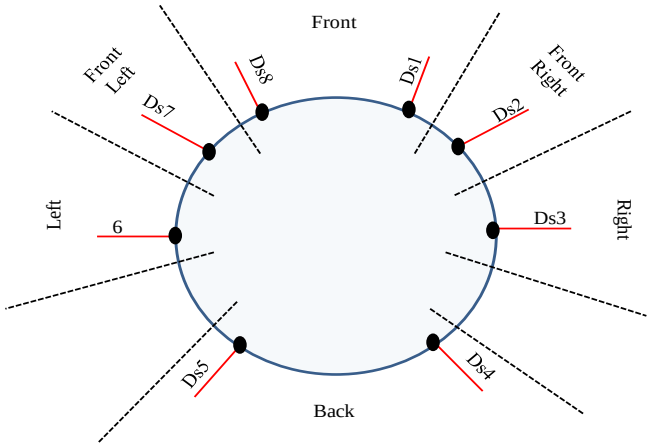



Figure 3.2. Schematic drawing of distance sensors in the E-puck robot.

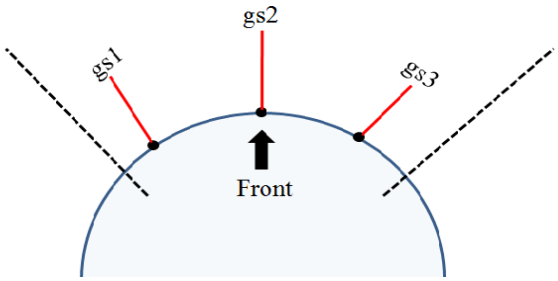



Figure 3.3. Ground sensors for the E-puck robot.

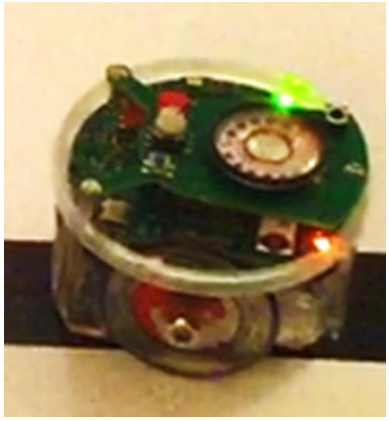


Figure 3.4. The real E-puck robot

# CHAPTER 4: TRAJECTORY PLANNING AND COLLISION AVOIDANCE ALGORITHM FOR MOBILE ROBOTICS SYSTEM

This section presents the results of research aimed to develop a new technique for line following and obstacle avoidance, relying on the use of infrared sensors. The sensors involves a reasonable level of calculations, so that it can be easily used in real-time control applications with microcontrollers [3].

## 4.1 Proposed Technique: Architecture and Design

In this brief, a fairly general technique is developed that has components of formation development, line follower and obstacles detection. The block diagram of the proposed technique is given in Figure 4.1 [3].

The controller receives input values directly from the infrared sensors. The robot controller applies the line follower (LFA) and collision avoidance (CAA) approaches. The (LFA) receives ground sensor readings as input values, then the controller will then issue a signal to the robot to adjust the motor speeds and follow the line; whereas the collision avoidance approach (CAA) receives distance sensor readings as an input value.

When an object is detected in front of the robot, CAA is responsible to spin the robot's direction and adjust its speed (according to the obstacle's position) in order to avoid collision. By applying both approaches, the robot follows the line and detects obstacles simultaneously. In other words, if an obstacle is detected, the robot must spin around the obstacle until it finds the line again [3].

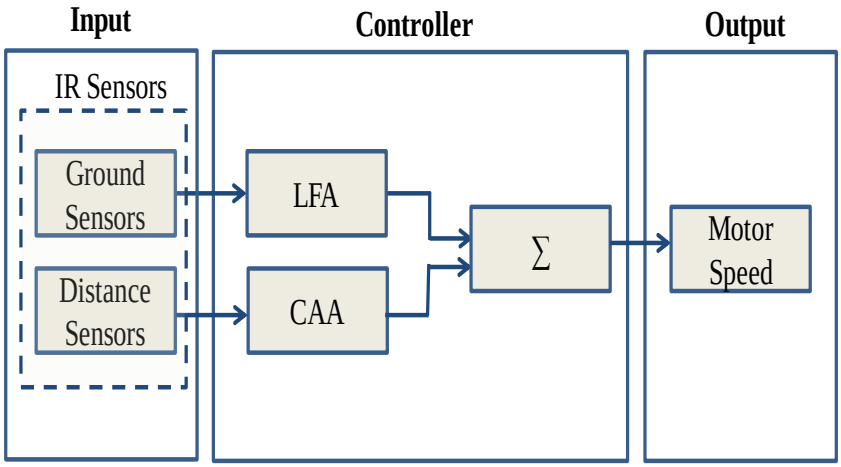



Figure 4.1. Block diagram of the proposed technique.

An efficient algorithm of the proposed technique is developed (as in Figure 4.2), to make the robot have the ability to follow the path and avoid obstacles along its way [3].

As shown in Figure 4.2, initialization is needed for the global variables that includes the number of distance and ground sensors used (8 distance sensors and 3 ground sensors) and the collision avoidance threshold value prior to starting the line follower and collision avoidance robot. After identifying the number of sensors used for each type, enabling these sensors is the next step [3].

For each time step of the simulation, all the eight distance sensors' and the three ground sensors' values are obtained. The three ground sensors are responsible for following the line; the motor speeds are adjusted based on these values. However, in case of possible collisions, the front distance sensors values are compared with a predefined collision avoidance threshold. If one of these sensors reaches the threshold, a front obstacle is detected. Subsequently, the readings of the right and left distance sensors are taken to determine the direction of the robot's movement in the case of a front obstacle. Later, it is compared with the threshold to check for any detected right/left obstacle. When the robot's path is determined, the left and right motor speeds are adjusted accordingly. After avoiding the obstacle, the robot will return to the line and continue following its path [3]. The flowchart is represented in Figure 4.3.

## 4.2 Simulation Setup

We show some simulations to illustrate the proposed approach as in Figure 4.4. All simulations were programmed using MATLAB as a controller. First, the robot starts sensing the environment with its three infrared sensors (ground sensors) and follows a black line drawn on the ground where there are two obstacles along its path. When the robot detects an object with its eight infrared sensors (distance sensors), it navigates around it to avoid collision based on the proposed algorithm stated above. Lastly, the robot recovers its path afterwards. The robot will follow the same steps each time it detects an obstacle and recovers its path again. The robot performance in different situations is illustrated by Figure 4.5 [3].

- Initialization of global variables (**TIME_STEP**: the time of one step of the simulation, **d**: number of distance sensors, **g**: number of ground sensors, **thr**: Collison avoidance threshold).
- Initialization of speed (**s**: global speed, **fs**: offset speed, **ls**: left motor speed, **rs**: right motor speed).
- **for** i = 1: d // enable distance sensors
- Get a unique identifier for each distance sensor.
- Enable all distance sensors every millisecond
- **end for**
- **for** i =1: g // enable ground sensors
- Get a unique identifier to each ground sensor.
- Enable all ground sensors every millisecond.
- **end for**
- **While (** TIME_STEP~= -1)
- **for** i =1: d // read distance sensor values
- ds (i) = get distance sensor readings.
- **end for**
- **for** i = 1: g // read ground sensor values
- gs (i) = get ground sensor readings
- **end for**
- **if** ((ds (1) > thr) ||( ds (8) > thr) ) // front obstacle is detected
- **then** stop the robot
- Take a reading of the right distance sensors (ds (2), ds (3), ds (4))
- Take a reading of the left distance sensors (ds (7), ds (6), ds (5))
- **if** ((ds (2)> thr) ||( ds (3)> thr) || (ds (4) > thr) ) // right obstacle is detected
- **then** ls - = s ; // turn left
  rs+ = s;
- **end if**
- **if** ((ds (7)>thr) || (ds (6) >thr)|| (ds (5)> thr)) //left obstacle is detected
- **then** ls + = s ; // turn right
  rs - = s;
- **end if**
- Set left and right motor speeds to ls & rs.
- Move forward
- Return to the line
- **else** // no obstacle is detected .
- //do line follower
- Δ= The difference between right and left ground sensors.
- ls=s-fs * Δ
- rs= s+f s * Δ
- Set left and right motor speeds to ls & rs.
- **end if**
- **end while**

Figure 4.2. Line follower and collision avoidance Algorithm.

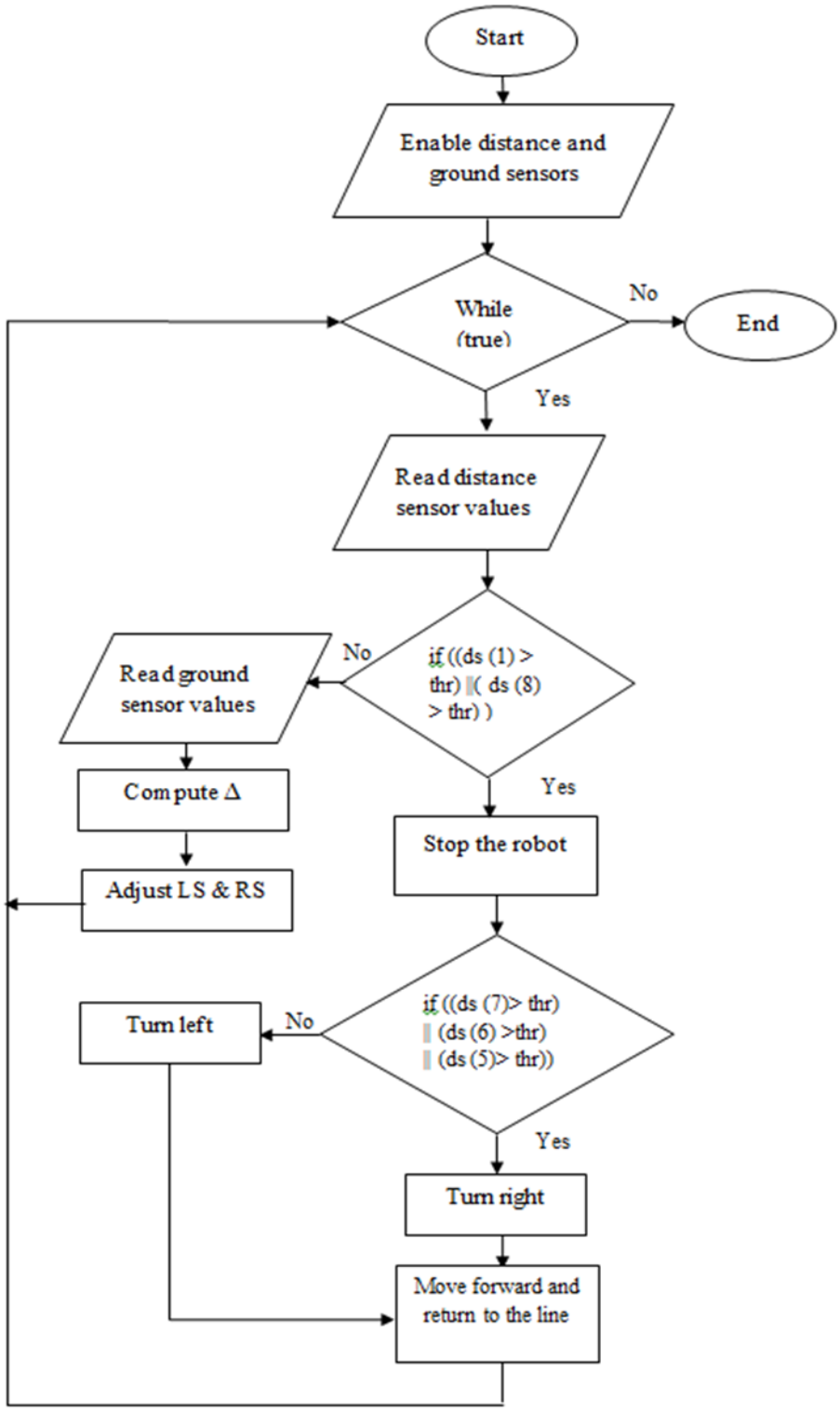



Figure 4.3.Flowchart of the proposed method.

Figure 4.5(a), shows the time of the simulation, where the robot detects the first object and stops. The robot checks both directions (left and right) to decide which way to proceed. It is similar to a  pedestrian crossing the roadway. The robot then gets the left sensor readings and right sensor readings in order to compare it with the threshold. Once these calculations are obtained, the robot will turn around the object to avoid obstacles as in Figure 4.5 (b), (c), and (d) [3].

As a final point, the robot successfully avoids the obstacles and recovers its path again as depicted in Figure 4.5(e). The robot will execute the same steps with each obstacle detected along its way as in Figure 4.5(f), (g), (h), and (i) [3].

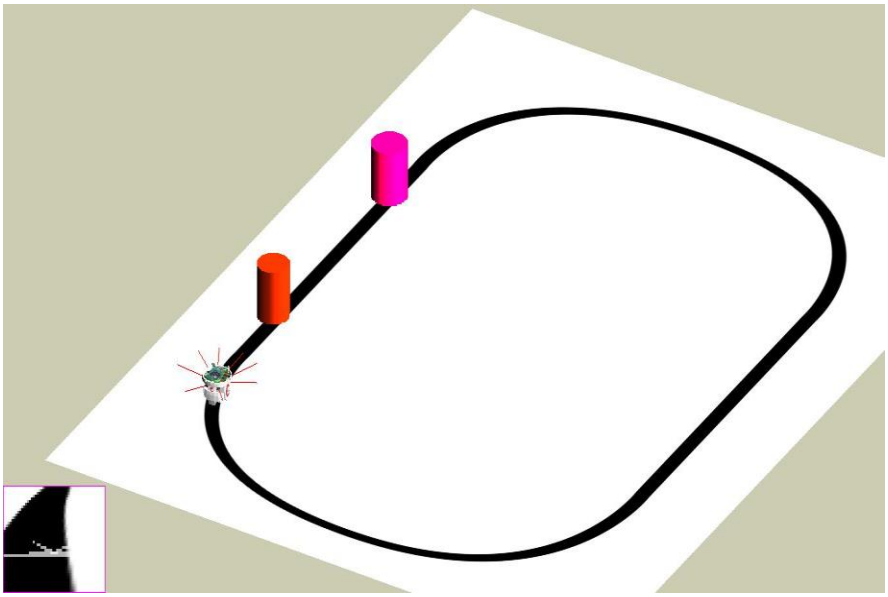


Figure 4.4. Robot platform of the simulation where there are two obstacles placed on the robot path

### 4.3 Results and Data Analysis

This section focuses on the results obtained from the simulation and the analysis of data gathered which validates both the performance and the effectiveness of the proposed technique. The following subsections discuss in detail the simulation results [3].

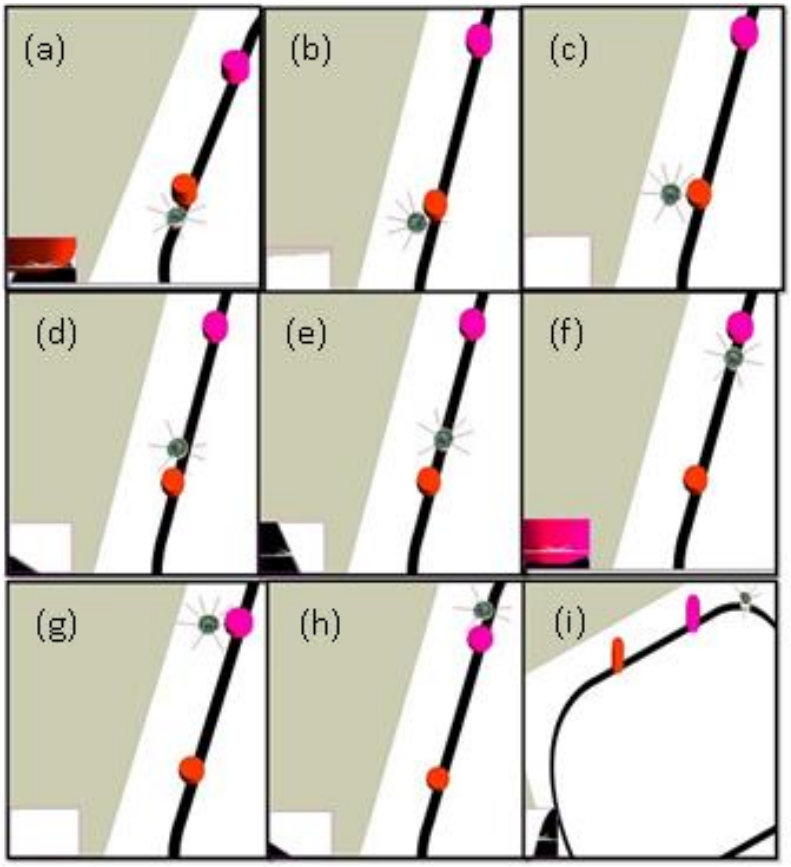


Figure 4.5. Snapshots of the simulation run at different simulation times

### 4.3.1 Distance Sensors Data Analysis

The E-puck is equipped with 8 distance sensors (Infrared sensors) for collision avoidance around the environment. These sensors have values varying from 0 to 2000, where 1000 or more means that the obstacle is very close, so the E-puck should avoid it accordingly. Various sensor measurements have been taken at different simulation times. Figure 4.6. shows the distance sensor readings at the beginning of the simulation (3 seconds of the simulation time) where all 8 distance sensors have low values (50 or less), due to no presence of obstacles along the robot path. However, once one of these 8 sensors detects an obstacle, their value increases to 1000 or more. As depicted in Figure

4.7, distance sensor 1 and 8 (the front sensors) have higher values than the remaining sensors (particularly, the left front sensor has more than 1000 value). This indicates that there is an obstacle in front of the robot [3].

According to the proposed algorithm, when one of the distance sensors reaches the threshold, the robot will avoid the obstacle and move around it to recover its path. Figure 4.8. and Figure 4.9., summarize the distance sensor readings once the robot returns to the line. It also shows that at 35 and 40 seconds of the simulation time, all sensor readings are low, which means that there is no obstacle in front or around the robot.

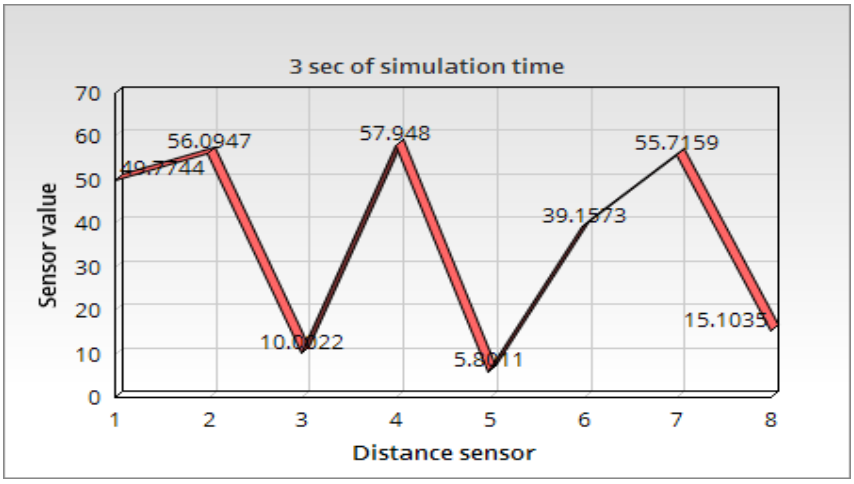



Figure 4.6. Distance sensors values at 3 seconds of the simulation time

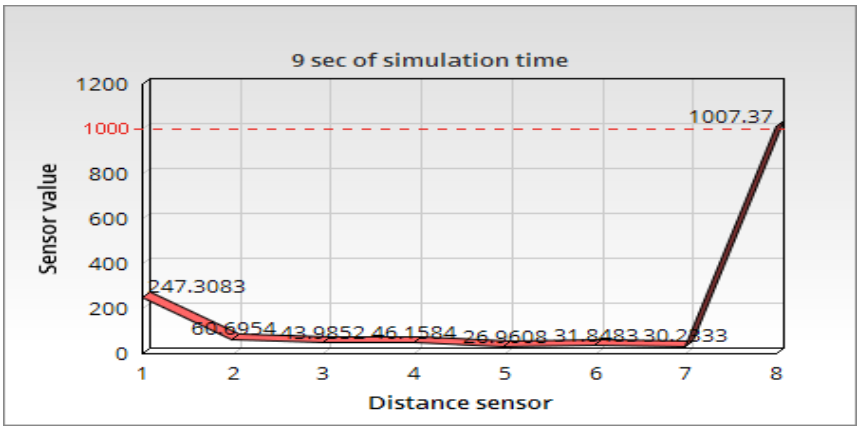



Figure 4.7. Distance sensors values at 9 seconds of the simulation time

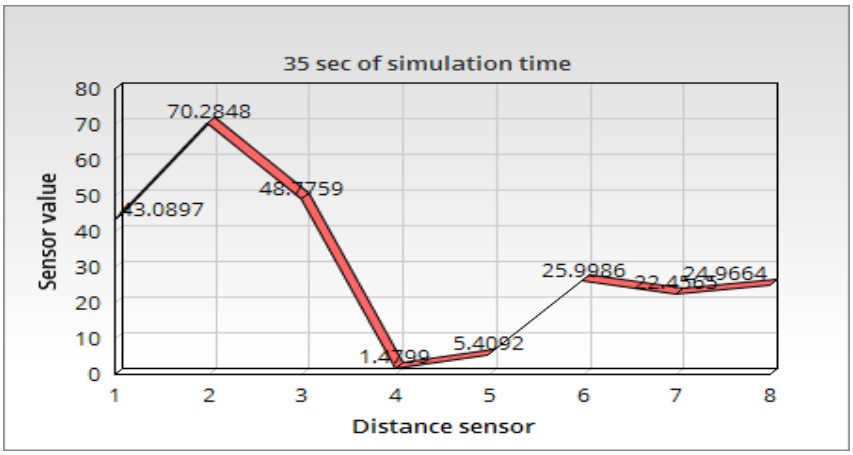



Figure 4.8. Distance sensors values at 35 seconds of the simulation time

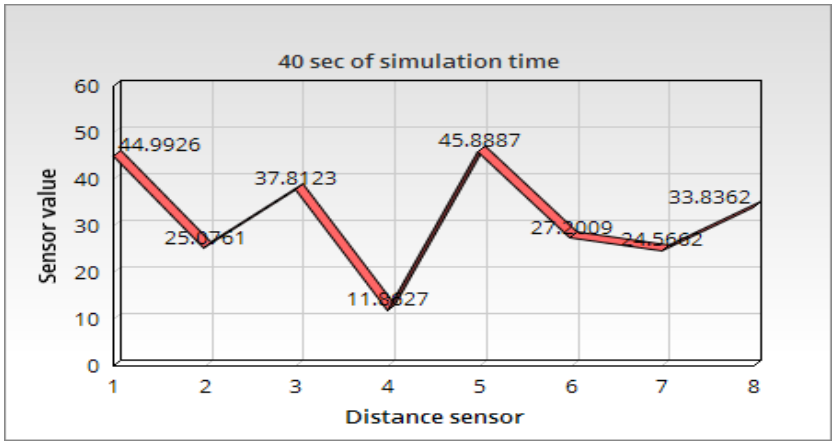



Figure 4.9. Distance sensors values at 40 seconds of the simulation time

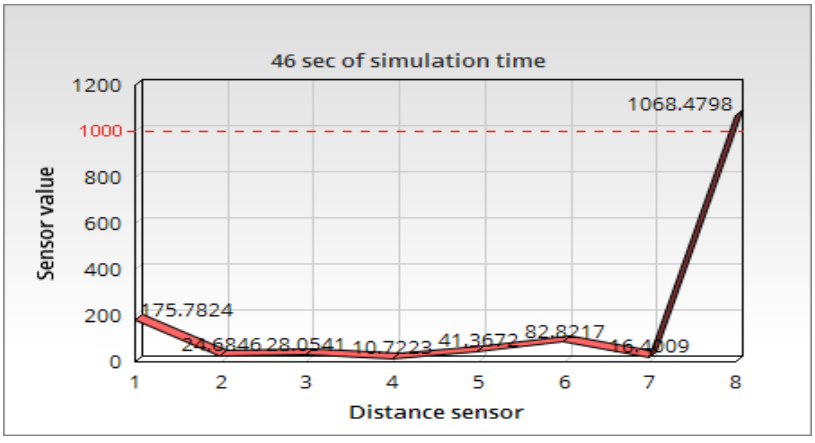



Figure 4.10. Distance sensors values at 46 seconds of the simulation time

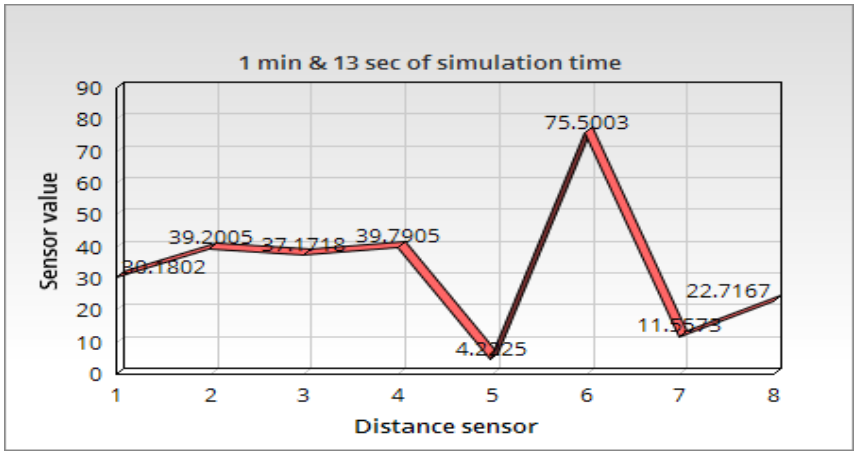



Figure 4.11. Distance sensors values at 1 min &13 sec of the simulation time

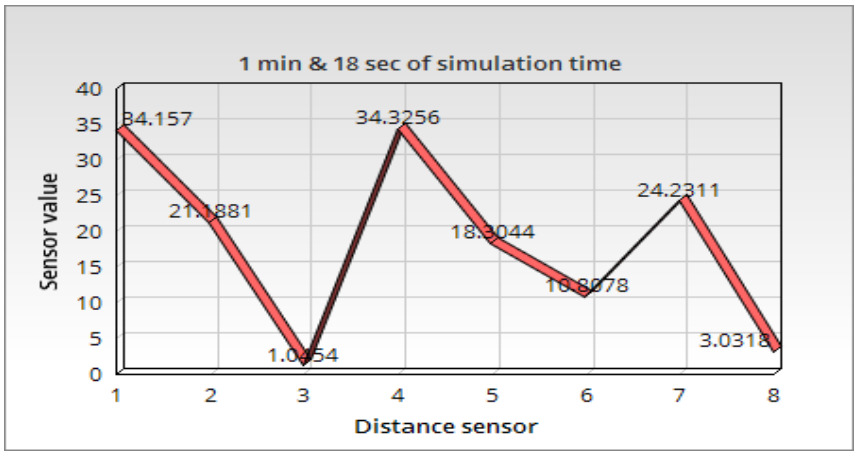



Figure 4.12. Distance sensors values at 1 min &18 sec of the simulation time

Table 4.1. Summarizes all distance sensors measurements at different simulation times.

| Distance sensor / Time/s | ds1 | ds2 | ds3 | ds4 | ds5 | ds6 | ds7 | ds8 |
|---|---|---|---|---|---|---|---|---|
| **3** | 49.7744 | 56.0947 | 10.0022 | 57.9480 | 5.8011 | 39.1573 | 55.7159 | 15.1035 |
| **9** | 247.3083 | 60.6954 | 43.9852 | 46.1584 | 26.9608 | 31.8483 | 30.2333 | 1007.3700 |
| **35** | 43.0897 | 70.2848 | 48.7759 | 1.4799 | 5.4092 | 25.9986 | 22.4565 | 24.9664 |
| **40** | 44.9926 | 25.0761 | 37.8123 | 11.8627 | 45.8887 | 27.2009 | 24.5662 | 33.8362 |
| **46** | 175.7824 | 24.6846 | 28.0541 | 10.7223 | 41.3672 | 82.8217 | 16.4009 | 1068.4798 |
| **1:13** | 30.1802 | 39.2005 | 37.1718 | 39.7905 | 4.2225 | 75.5003 | 11.5573 | 22.7167 |
| **1:18** | 34.1570 | 21.1881 | 1.0454 | 34.3256 | 18.3044 | 10.8078 | 24.2311 | 3.0318 |

At 46 seconds of the simulation time, another obstacle is detected with one of the front sensors which reaches the threshold as represented in Figure 4.10. In Figure 4.11. and 4.12., the sensors measurements are shown at 1 minute and 13 seconds, and 1 minute and 18 seconds respectively. Both figures indicate low sensor readings thus there are no obstacles detected. In addition, Table 4.1 summarizes all 8 distance sensor values at various simulation times [3].

### 4.3.2 Ground Sensors Data Analysis

The E-puck robot is also equipped with three ground sensors which are infrared sensors facing the ground. Their role is to detect the black line in a white surface in order to guide the robot. After obtaining the ground sensor readings, Δ is computed, which is the difference between the right and left ground sensors. After this, the left and right motor speeds are adjusted accordingly. Table 4.2 shows all three ground sensor values at several simulation times along with their Δ values [3].

Table 4.2. Summarizes all ground sensors readings and Δ values at different simulation times.

| Time/s | gs1 | gs2 | gs3 | Δ |
|---|---|---|---|---|
| **3** | 304.6110 | 280.0890 | 311.4296 | 6.8186 |
| **9** | 333.6138 | 295.8490 | 312.3697 | -21.2441 |
| **35** | 242.8890 | 287.1930 | 962.9207 | 720.0316 |
| **40** | 273.9332 | 343.2757 | 301.9313 | 27.9981 |
| **46** | 292.1894 | 303.0826 | 333.1293 | 40.9399 |
| **1:13** | 262.5687 | 246.0613 | 976.2003 | 713.6317 |
| **1:18** | 652.2564 | 286.8184 | 323.2484 | -329.008 |

Figure 4.13 displays all three ground sensor measurements during the simulation whereas Figure 4.14 shows the left and right robot's motor speeds. As indicated in these two figures, at 35 s, 1:13 s, and 1:18 s, the highest values of the ground sensors and speeds are recorded. At 35 s and 1:13 s, where the robot detects the first and second obstacles, the right ground sensor (gs3) has a very high value, which indicates that the robot is turning sharply left to avoid the obstacles (Figure 4.13) [3].

Furthermore, Figure 4.14 validates this outcome as shown at 35 s and 1:13 s of the simulation time. The left motor speeds are in negative values, whereas the right motor speeds are in very large positive values. This demonstrates that the robot is turning sharply left to avoid the obstacle in front of it [3].

Finally, as shown in Figure 4.13, at 1:18 s, the left ground sensor (gs1) is reading a much higher value than the right ground sensor (gs3), because the robot is turning right due to the curvy line (see also Figure 4.5(i)). In addition, Figure 4.14 shows that at 1:18 s, the left motor speed is much higher as compared to the right motor speed which indicates again that the robot is turning right to follow the curvy line [3].

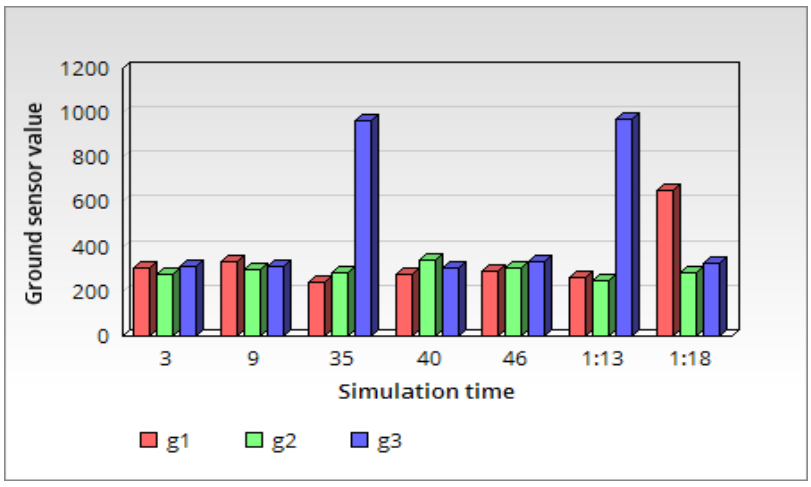



Figure 4.13. All three ground sensors measurements at different simulation times.

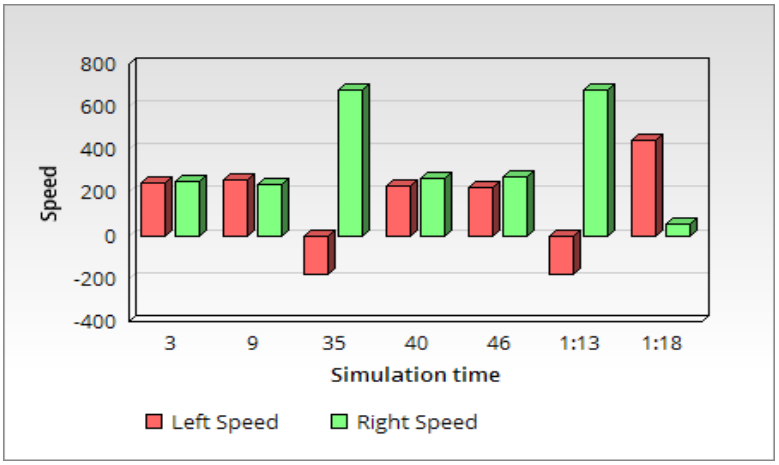



Figure 4.14. Left and right motor speeds

# CHAPTER 5: SENSOR FUSION BASED MODEL FOR COLLISION FREE MOBILE ROBOT NAVIGATION

## 5.1 Introduction

The area of autonomous mobile robot has gained an increasing interest in the last decade. Autonomous mobile robots are robots that can navigate freely without human involvement. There are many sensor fusion techniques that have been proven to be effective and beneficial especially in detecting and avoiding obstacles, as well as path planning of the mobile robot. Fuzzy logic, neural network, neuro-fuzzy, and genetic algorithms are examples of well-known fusion techniques that help in moving the robot from the starting point to the target without colliding with any obstacles along its path [4].

Many researchers have used sensor fusion to fuse data from various types of sensors, which improved the decision making process of routing the mobile robot. A hybrid mechanism was introduced by [92], which uses the neuro-fuzzy controller for collision avoidance and path planning behavior for mobile robots in unknown environments. Moreover, an Adaptive Neuro-fuzzy Inference System (ANFIS) was

applied for an Autonomous Ground Vehicle (AGV) to safely reach the target while avoiding obstacles by using four ANFIS controllers [93]. Another sensor fusion based on Unscented Kalman Filter (UKF) was used for mobile robots' localization problems. Accelerometer, encoders, and gyroscope were used to obtain data for the fusion algorithm. The proposed work was tested experimentally and was successfully capable of tracking the motion of the robot [94]. In [95], Teleoperated Autonomous Vehicle (TAV) was designed with collision avoidance and path following techniques to discover the environment. TAV includes GPS, infrared sensors, and the camera. Behavior based architecture is proposed, which consists of Obstacle Avoidance Module (OAM), Line Flowing Module (LFM), Line Entering Module (LEM), Line Leaving Module (LLM) and U-Turn Module (UTM). Sensor fusion based on fuzzy logic was used for collision avoidance while neural network fusion was used for the line following approach.

This section focuses on the integration of multisensory information from range finder camera and infrared sensors using fuzzy logic fusion system for collision avoidance and line follower mobile robot. The proposed methodology develops membership functions for inputs and outputs and designs fuzzy rules based on these inputs and outputs [4].

## 5.2 Proposed Methodology and Design of the Fusion Model

This section presents the proposed methodology for mobile robot collision-free navigation with the integration of the fuzzy logic fusion technique. The mobile robot is equipped with distance sensors, ground sensors, camera, and GPS. Distance sensors which are infrared sensors, and the camera are used for collision avoidance behavior

where the ground sensors are used for path following behavior. GPS is used to get the robot's position. The goal of the proposed technique is as follows [4]:

- The capability of the mobile robot to avoid obstacles along its path.
- The integration of sensor fusion using fuzzy logic rules based on sensor inputs and defined membership functions.
- The capability of the mobile robot to follow a predetermined path.
- The performance of the mobile robot when programmed with the fuzzy logic sets and rules.

Multisensory fusion model is designed for better obstacle detection and avoidance, by fusing eight distance sensors and the range finder camera. The fusion model is based on fuzzy logic fusion technique using MATLAB software. A Fuzzy Logic System (FLS) is composed of four main parts which are: fuzzifier, rules, inference engine, and defuzzifier. The block diagram of FLS is shown in Figure 5.1 [4].

The fuzzification stage is the process of converting a set of inputs to fuzzy sets based on defined fuzzy variables and membership functions. According to a set of rules, the inference is made. Finally, at the defuzzification stage, membership functions are used to map every fuzzy output to a crisp output [4].

### 5.2.1 Fuzzy Sets of the Input and Output

There are nine inputs to the fuzzy logic system and two outputs. The inputs are basically the values of eight distance sensors donated as SF1, SF2, SR1, SR2, SL1, SL2, SB1, and SB2. These sensors measure the amount of light in a range of 0 to 2000 where

the threshold is set to 1000 for detected obstacles. The ninth input is the range finder camera value that measures the distance to an obstacle. Two outputs are Generated Left Velocity (LV) and Right Velocity (RV). Figure 5.2 shows the Mamdani System using Fuzzy Inference System (FIS) with nine inputs and two outputs [4].

### 5.2.2 Membership Functions of the Input and Output

Input variables of distance sensors readings are divided into membership functions which are Obstacle not found (OBSNF), and Obstacle found (OBSF). Both membership functions are a type of trapezoidal-shaped membership function [4].

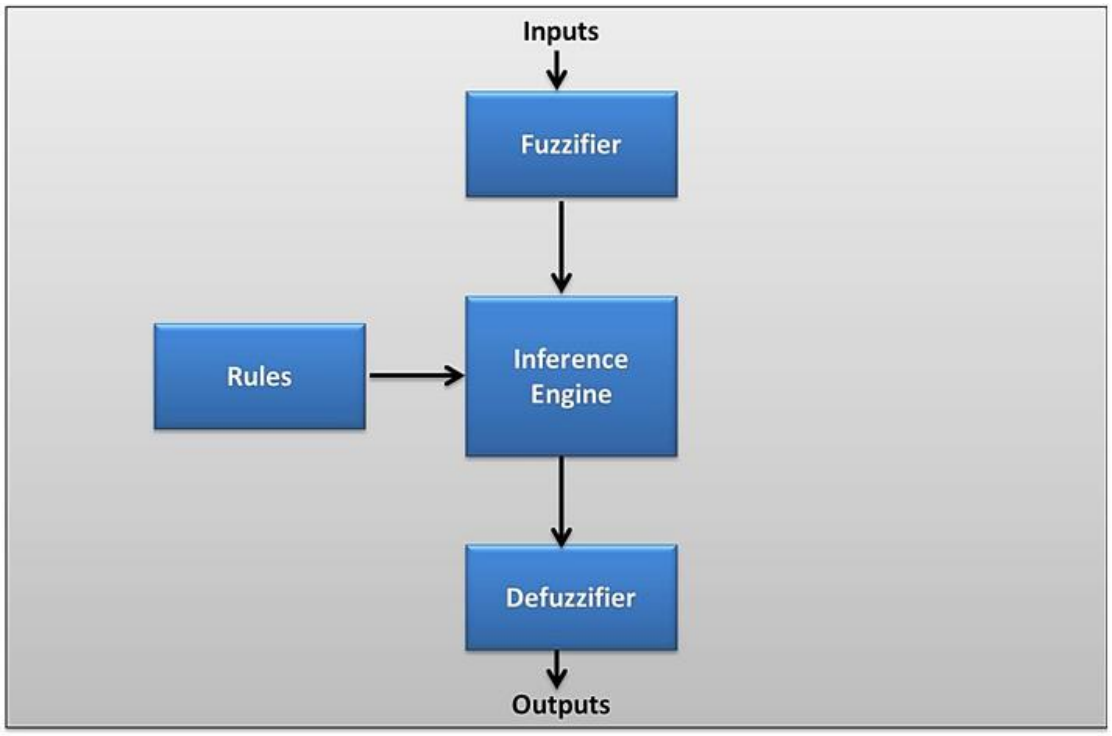



Figure 5.1. Block diagram of the Fuzzy Logic System.

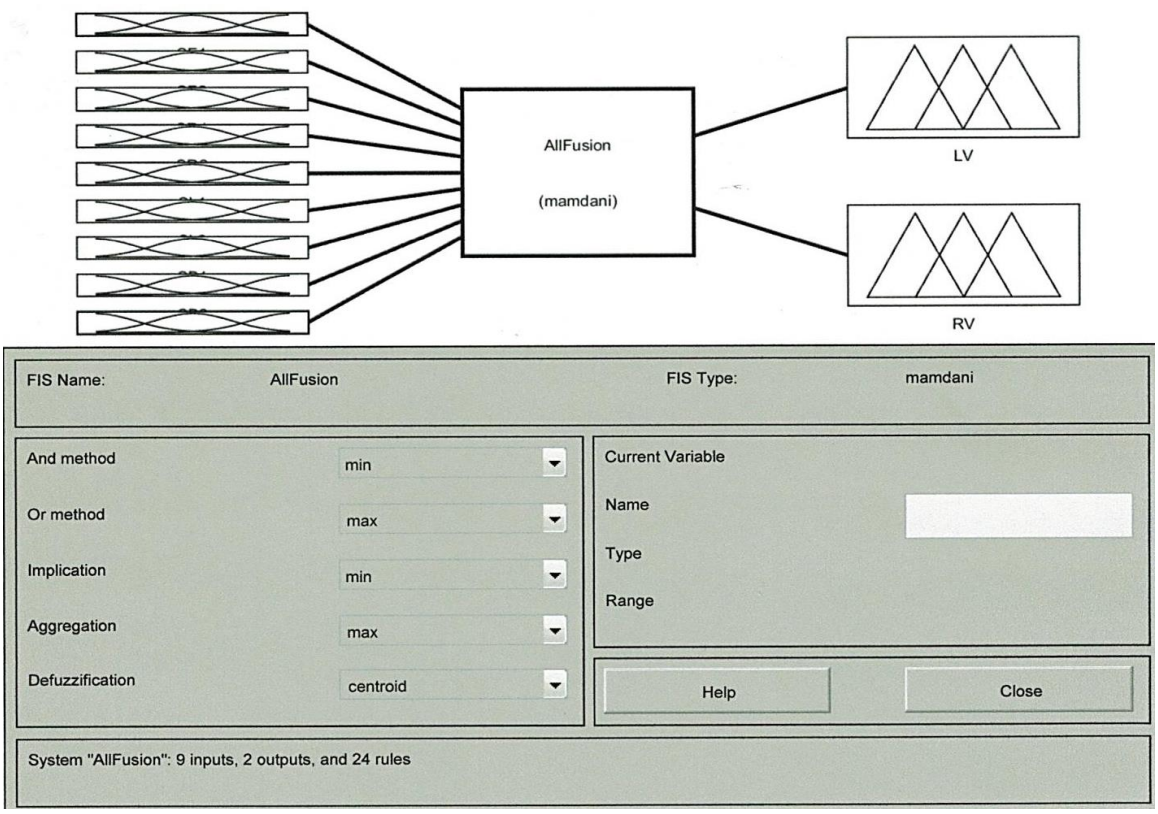



Figure 5.2. Fuzzy Inference System (FIS) with inputs and outputs.

The range for the distance sensor values is [0-2000] and the threshold is set to 1000, where the value of 1000 or more means an obstacle is found and the robot should avoid it. The input variables of the range finder camera are divided into two trapezoidal-shaped membership functions "Near" and "Far". The range finder camera measures the distance from the camera to an obstacle in meters. The overall range of the camera input is [0 1] where 0.1 m is considered as "Near" distance, and collision behavior avoidance should be applied. The input membership functions for distance sensors and the camera are displayed in Figures 5.3 and 5.4, respectively [4].

Let's assume that $x$ is the sensor value and $R$ is the range of all sensors values where $x \in R$. The trapezoidal-shaped membership function based on four scalar parameters $i$, $j$, $k$, and $l$, can be expressed as in (5.1) [4].

$$\mu_{trap}(x: i, j, k, l) = max(min\left(\frac{x-i}{j-i}, 1, \frac{l-x}{l-k}, 0\right)) \quad (5.1)$$

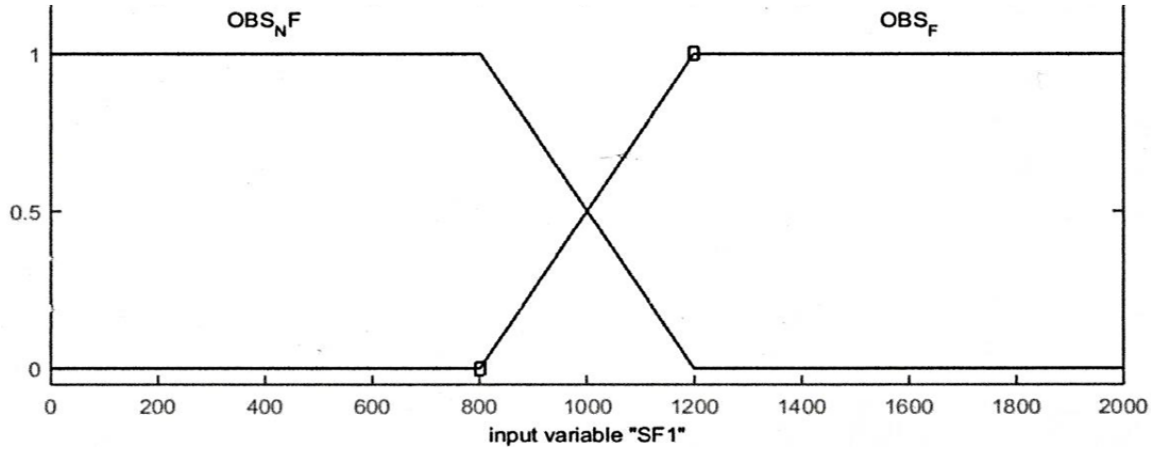



Figure 5.3. Input membership functions for the distance sensors

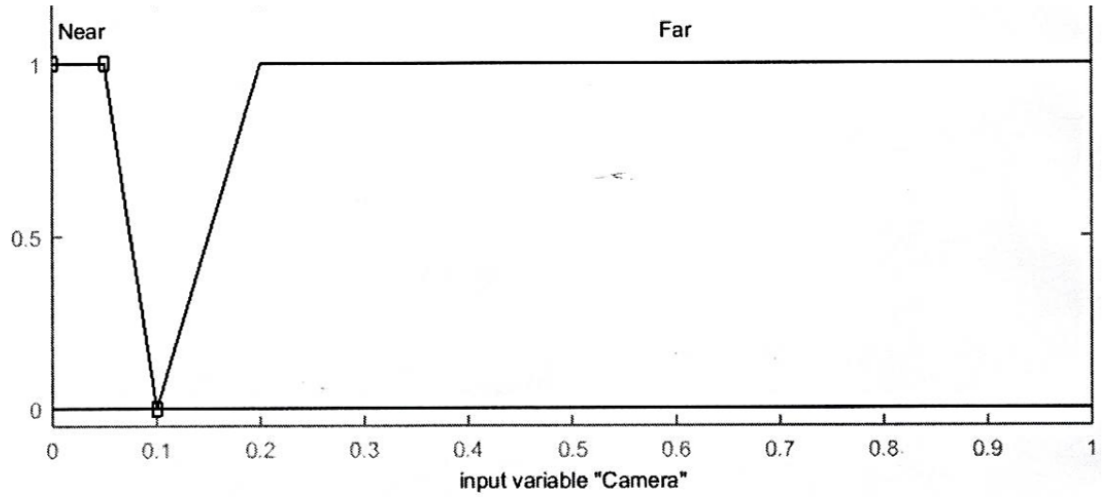



Figure 5.4. Input membership functions for the camera

The output variables of left and right velocities of the mobile robot (LV and RV) are divided into two membership functions negative velocity "NEG_V" and positive velocity "POS_V". The effect and the action of these two memberships on the differential wheels of the robot are summarized as follows [4]:

- If both LV and RV speeds are set to POS_V, then the robot will move forward.

- If LV is set to POS_V and RV is set to NEG_V, then the robot will turn right.
- If LV is set to NEG_V and RV is set to POS_V, then the robot will turn left.

The “NEG_V” is a Z-shaped membership function. This function is represented in (5.2) where $u$ and $q$ are two parameters of the most left and most right of the slope [4].

$$\mu_z(x) = \begin{cases} 1, & x \le u \\ 1 - 2\left(\frac{x-u}{q-u}\right)^2, & u \le x \le \frac{u+q}{2} \\ 2\left(\frac{x-q}{q-u}\right)^2, & \frac{u+q}{2} \le x \le q \\ 0, & x \ge q \end{cases} \tag{5.2}$$

In addition, the “POS_V” is an S-shaped membership function where *y1*and *y2* are two parameters of the leftmost and rightmost of the slope. The S-shaped membership function can be expressed as in (5.3). Figure 5.5 shows the output membership functions [4].

$$\mu_s(x) = \begin{cases} 1, & x \le y1 \\ 2\left(\frac{x-y1}{y2-y1}\right)^2, & y1 \le x \le \frac{y1+y2}{2} \\ 1 - 2\left(\frac{x-y2}{y2-y1}\right)^2, & \frac{y1+y2}{2} \le x \le y2 \\ 0, & x \ge y2 \end{cases} \tag{5.3}$$

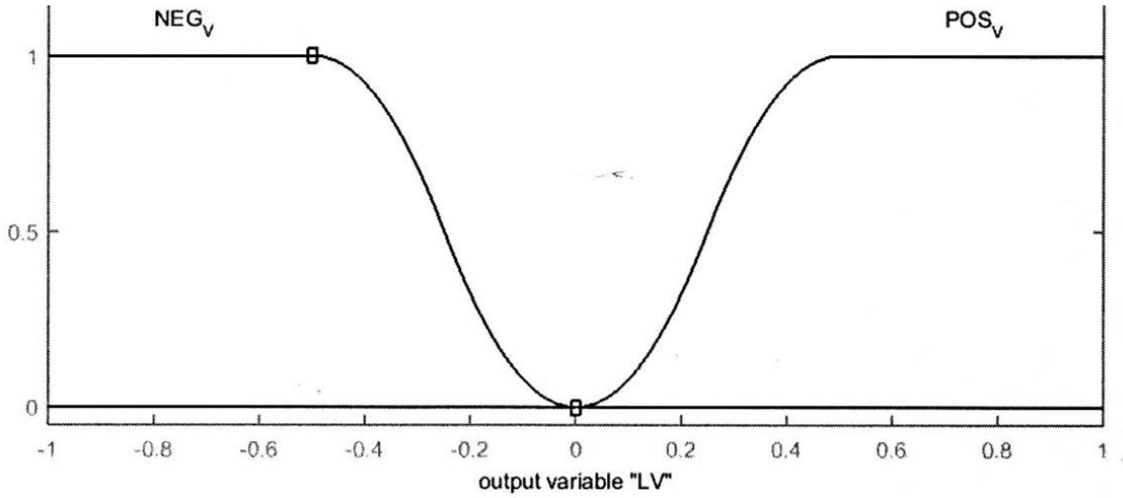



Figure 5.5. Output membership functions.

### 5.2.3 Designing Fuzzy Rules

Based on the membership functions of the fuzzy sets and inputs and outputs, rules are defined. There are 24 rules for collision avoidance of the mobile robot. Figure 5.6 shows an example of the rule editor in MATLAB. We can use AND or OR operation for connecting membership values, where the fuzzy AND is the minimum of two or more membership values and OR is the maximum of two or more membership values. Let $\mu\gamma$ and $\mu\delta$ be two membership values, then the fuzzy AND and fuzzy OR are described as in Equation (5.4), and (5.5) respectively. In addition, Table 5.1 lists all the rules with the fuzzy AND operator that expresses the movement behavior of the mobile robot [4].

$$\mu_\gamma \text{ AND } \mu_\delta = \min(\mu_\gamma, \mu_\delta) \tag{5.4}$$

$$\mu_\gamma \text{ OR } \mu_\delta = \max(\mu_\gamma, \mu_\delta) \tag{5.5}$$

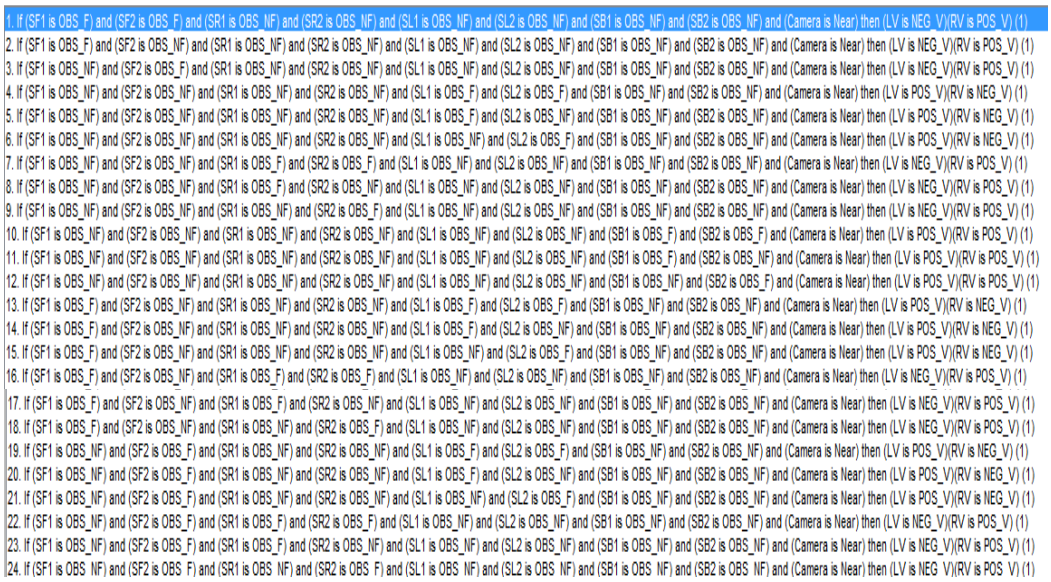



Figure 5.6. Example of the rule editor in MATLAB

Table 5.1. Fuzzy logic rules.

| No | SF1 | SF2 | SR1 | SR2 | SL1 | SL2 | SB1 | SB2 | LV | RV |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | OBS_F | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 2 | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 3 | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 4 | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_F | OBS_NF | OBS_NF | POS_V | NEG_V |
| 5 | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | POS_V | NEG_V |
| 6 | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | POS_V | NEG_V |
| 7 | OBS_NF | OBS_NF | OBS_F | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 8 | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 9 | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 10 | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_F | POS_V | POS_V |
| 11 | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_NF | POS_V | POS_V |
| 12 | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_F | POS_V | POS_V |
| 13 | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_F | OBS_NF | OBS_NF | POS_V | NEG_V |
| 14 | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | POS_V | NEG_V |
| 15 | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | POS_V | NEG_V |
| 16 | OBS_F | OBS_NF | OBS_F | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 17 | OBS_F | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 18 | OBS_F | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 19 | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_F | OBS_F | OBS_NF | OBS_NF | POS_V | NEG_V |
| 20 | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | POS_V | NEG_V |
| 21 | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_F | OBS_NF | OBS_NF | POS_V | NEG_V |
| 22 | OBS_NF | OBS_F | OBS_F | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 23 | OBS_NF | OBS_F | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |
| 24 | OBS_NF | OBS_F | OBS_NF | OBS_F | OBS_NF | OBS_NF | OBS_NF | OBS_NF | NEG_V | POS_V |

### 5.2.4 Defuzzification

The last step of designing the fuzzy logic fusion system is the defuzzification process where outputs are generated based on fuzzy rules, membership values, and a set of inputs. The method used for defuzzification is the Centroid method [4].

Moreover, the fuzzy logic fusion model was designed for preventing the mobile robot from colliding with any obstacles while following the line. The fusion model composed of nine inputs, two outputs, and twenty four rules. Figure 5.7 demonstrates the proposed methodology. As shown in Figure 5.7, the initialization of the robot and its sensors is the first step. After that, the distance sensors and camera values are fed into the fuzzy logic fusion system for obstacles detection and distance measurements. If an obstacle is found, the mobile robot will adjust its speed for turning left or right based on the position of the obstacle. The decision is made based on defined fuzzy rules. After avoiding the obstacle, the mobile robot should continue following the line by obtaining ground sensor values and finally adjust its speed accordingly. On the other hand, if there is no obstacle detected, the mobile robot should follow the line while it checks for obstacles to avoid at each time step [4].

## 5.3 Simulation and Real-Time Implementation for Mobile Robot Navigation

Webots Pro simulator is used to model the robot (E-puck) and the environment. Webots Pro simulator and the E-puck have been described previously in chapter 3. However, a camera and GPS have been added to the E-puck robot in this work.

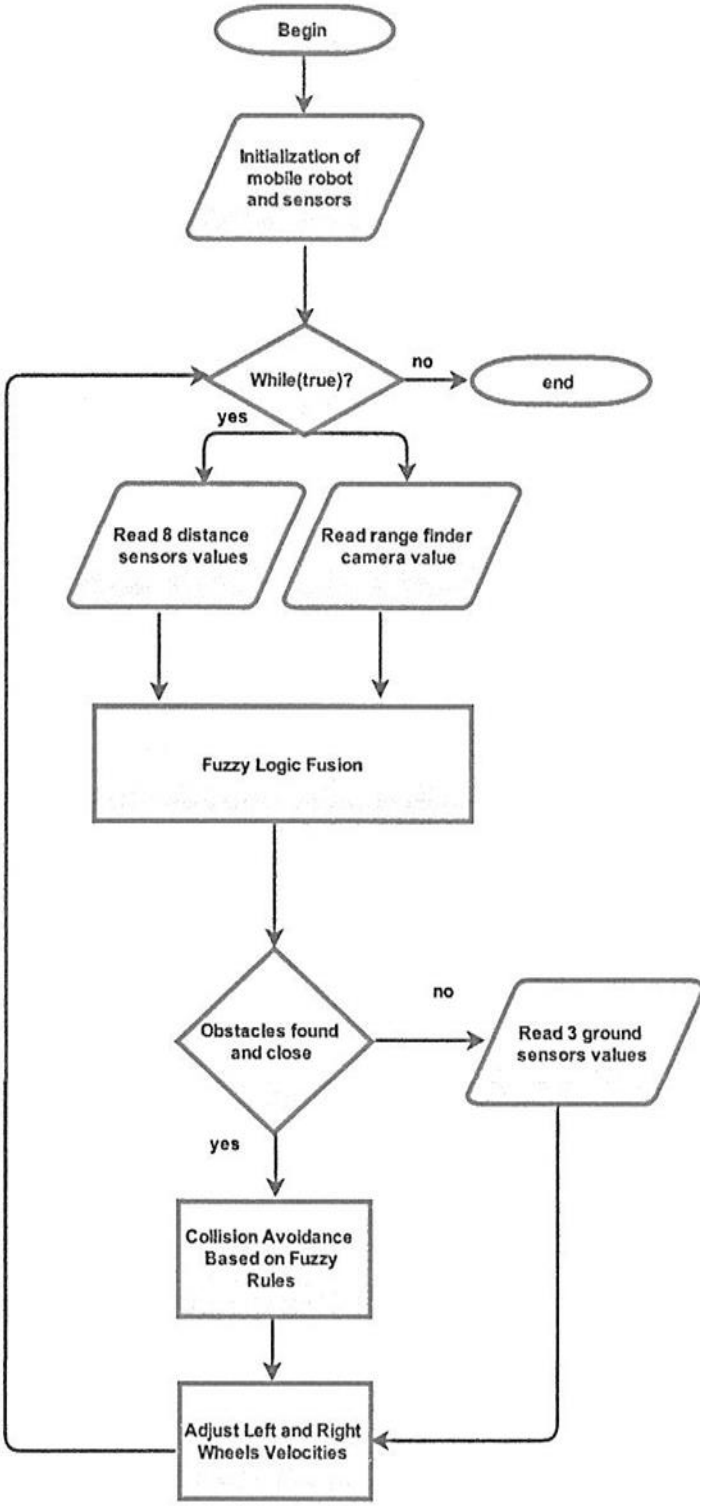



Figure 5.7. Flowchart of the proposed methodology.

The camera sensor that is used in this work is a range finder type of camera which allows obtaining distance in meters between the camera and the obstacle from the OpenGL context of the camera. GPS is used to get the position of the robot. Figure 5.8, shows the E-puck robot top view with the range finder camera, the distance sensors, and the ground sensors [4].

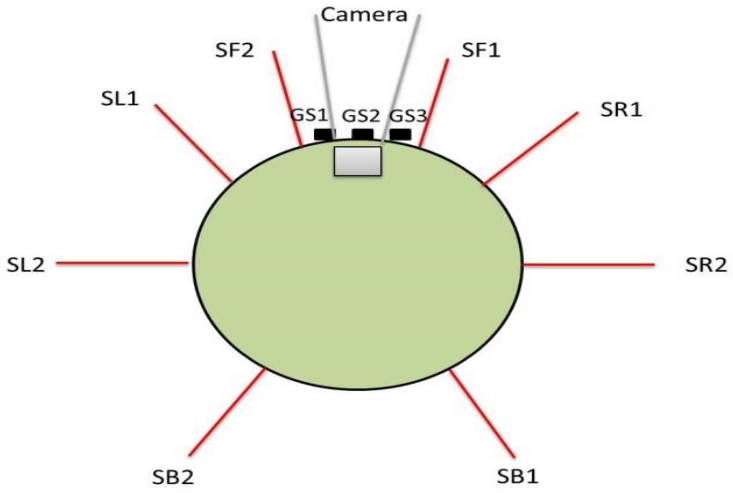



Figure 5.8. The E-puck robot with various types of sensors.

The E-puck used has eight distance sensors which are infrared sensors, camera, three ground sensors, and GPS. The E-puck first senses the environment for possible collisions by using the distance sensors and the range finder camera readings. If there is no obstacle detected, the E-puck follows a black line drawn on a white surface. Snapshots of the simulation and real-time experiment for one robot detecting and avoiding an obstacle while following the line are depicted in Figures 5.9 and 5.10, respectively. Both figures show the environment with one mobile robot moving forward until it detects an obstacle. After the detection of the obstacle, all readings are fed into the proposed fuzzy logic fusion model, and based on the defined fuzzy rules, the E-puck will turn

accordingly by adjusting the left and right wheels velocities. After that, the E-puck will continue moving forward and follow the line [4].

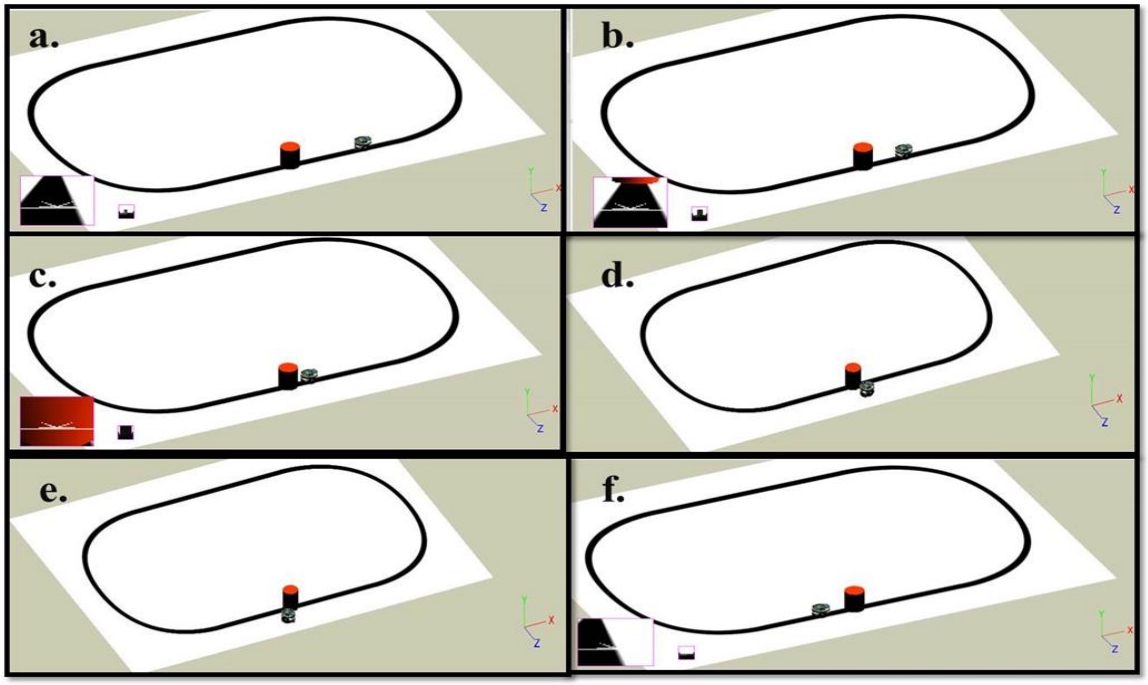



Figure 5.9. Simulation snapshots of one robot and one obstacle.

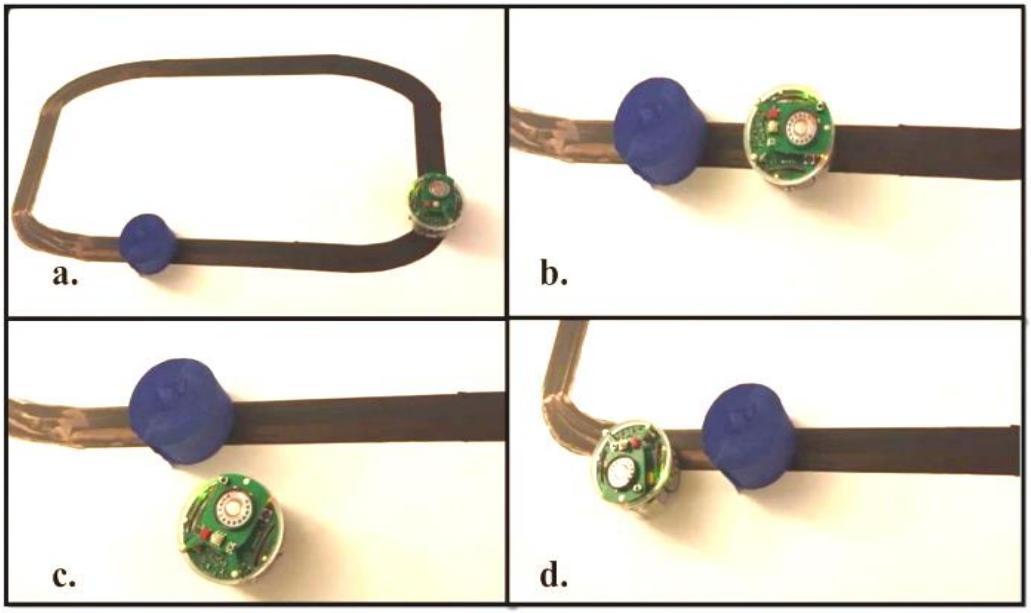



Figure 5.10. Snapshots of the real-time experiment of one robot and one obstacle.

In addition, a more complex environment with various obstacles in different shapes and sizes has been modeled and tested through simulation and real-time experiments. Figures 5.11 and 5.12 present snapshots of the simulation and real-time experiments for two robots following a black line and avoiding different types of obstacles. As shown in these figures, both robots face and detect each other successfully. Each robot tries to avoid the other by adjusting its speeds and returns to follow the line. These robots are considered as dynamic obstacles to each other [4].

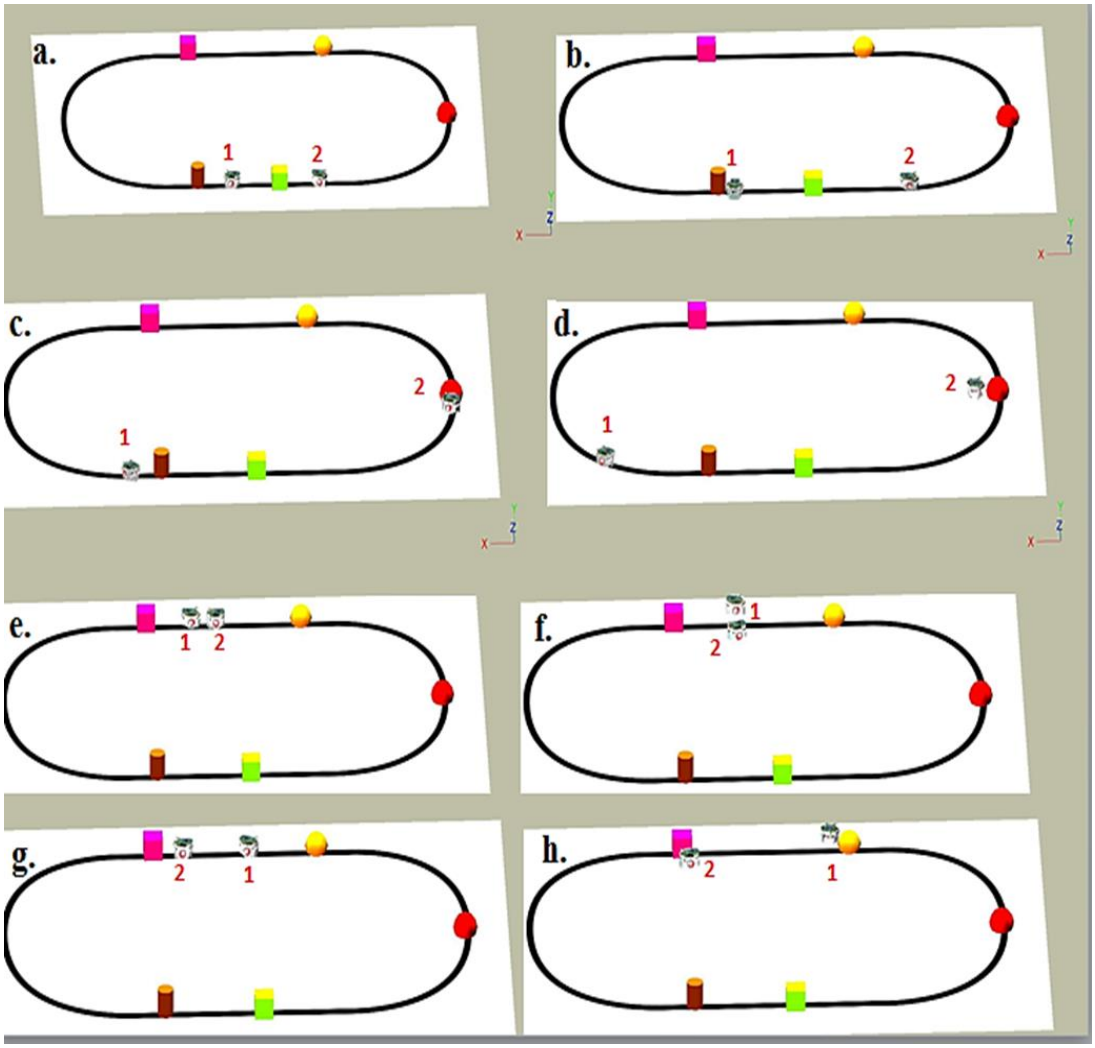



Figure 5.11. Simulation snapshots for multiple robots and obstacles

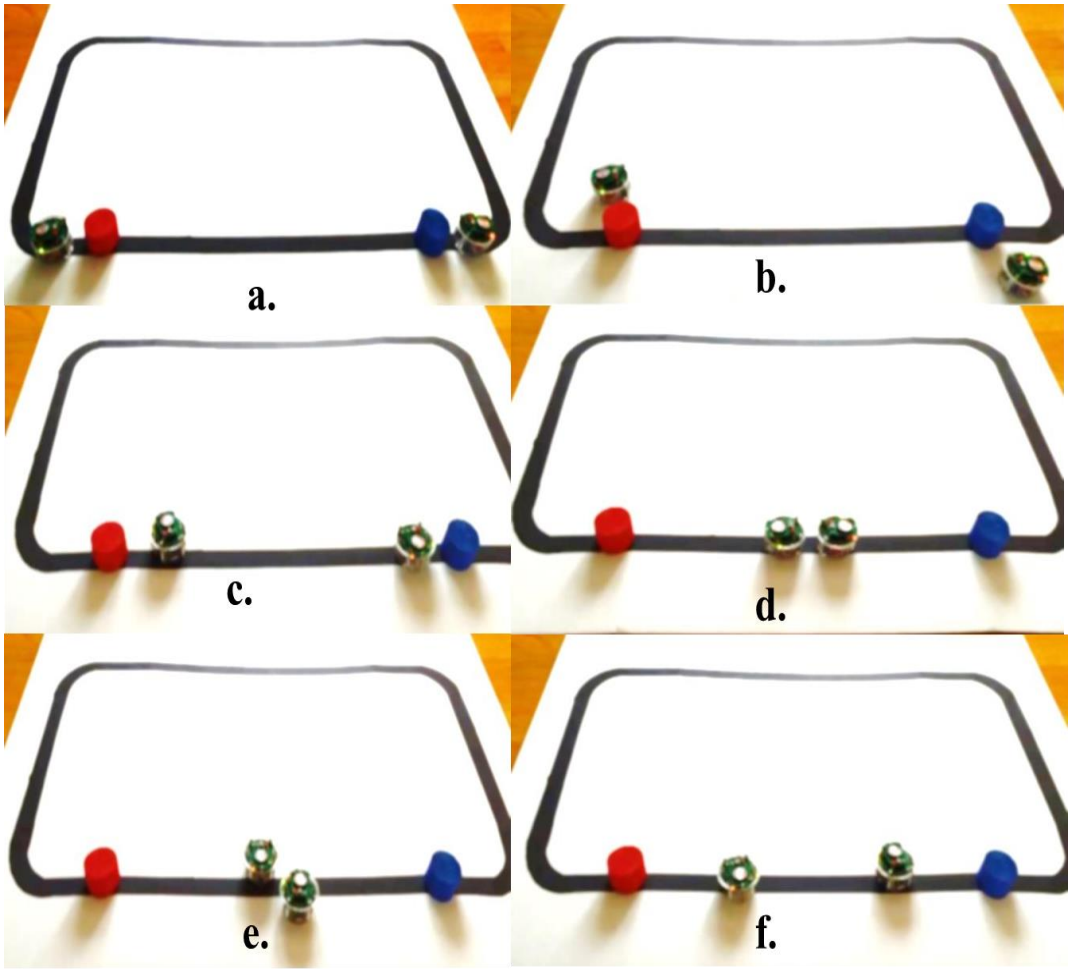


Figure 5.12. Real-time experiment for multiple robots and obstacles.

## 5.4 Results and Investigation of the Proposed Model

This section presents the sensors values obtained from the simulation at different time steps. Three different scenarios have been presented. The first one is a simple environment containing one obstacle and one robot, whereas the second one is a more complex environment that has more static and dynamic obstacles with two robots. The third scenario has more cluttered obstacles, which makes it a more challenging environment for the mobile robot navigation [4].

### 5.4.1 First Scenario

Tables 5.2, and 5.3 show the sensors values before and after applying the fuzzy logic fusion model in a simple environment at three different simulation times. At T1, the robot is far away from the obstacle; at T2, the robot is very close to the obstacle, and at T3, the robot has passed the obstacle successfully. When distance sensor values are below the threshold (1000), it means that there is no obstacle detected. However, when it goes above the threshold, it means that there is an obstacle and the robot needs to adjust its movement. As shown in Table 5.2, the front distance sensor SF1 has a value of 1159.18, which is higher than the threshold value at T2, and it occurs before applying the fuzzy logic fusion method [4].

In addition, both front distance sensors SF1 and SF2 have higher values than the threshold, which are 1127.19 and 1077.76, respectively, at T2 where the fuzzy logic technique is applied. Again, this means that there is an obstacle detected. Table 5.3 shows the distance to obstacles measured in meters by the range finder camera at various simulation times: T1, T2, and T3. As represented in Table 5.3, once the robot approaches the obstacle, the distance between the robot and the obstacle is decreased. At T2, the distance between the robot and the obstacle, where the obstacle is firstly detected by the camera and before applying the fuzzy logic method is 0.097 m; where it is only 0.040 m after applying the fusion model. The camera can measure the distance up to one meter ahead. At T3, the camera could not measure the distance to an obstacle, because it did not find any obstacles within its range [4].

Table 5.2. Distance sensors values before and after implementing the fusion model.

| Distance Sensor | Without Fuzzy Logic Fusion | | | With Fuzzy Logic Fusion | | |
|---|---|---|---|---|---|---|
| | T1=5 | T2=15 | T3=41 | T1=5 | T2=22 | T3=41 |
| SF1 | 14.71 | 1159.18 | 64.05 | 11.73 | 1127.19 | 72.72 |
| SF2 | 35.19 | 220.33 | 53.72 | 48.48 | 1077.76 | 54.80 |
| SR1 | 59.53 | 24.41 | 40.20 | 45.69 | 35.14 | 42.40 |
| SR2 | 31.55 | 26.77 | 4.68 | 31.54 | 28.00 | 28.47 |
| SL1 | 23.78 | 36.03 | 59.58 | 21.23 | 58.54 | 31.90 |
| SL2 | 59.49 | 18.59 | 33.88 | 13.40 | 38.10 | 72.51 |
| SB1 | 19.99 | 91.65 | 14.64 | 22.21 | 56.23 | 59.42 |
| SB2 | 46.62 | 13.29 | 5.08 | 66.13 | 30.13 | 47.91 |

Table 5.3. Distance measurements by camera before and after implementing the fusion model.

| Range Finder Camera | Without Fuzzy Logic Fusion | | | With Fuzzy Logic Fusion | | |
|---|---|---|---|---|---|---|
| | T1=5 | T2=13 | T3=41 | T1=5 | T2=22 | T3=41 |
| Distance to obstacle in meter | 0.337 | 0.097 | x | 0.339 | 0.0404 | X |

Table 5.4. Ground sensors values at different simulation times

| Simulation Time | Ground sensor 1 | Ground sensor 2 | Ground sensor 3 | Delta |
|---|---|---|---|---|
| 5 | 287.83 | 256.58 | 330.61 | 42.77 |
| 10 | 271.00 | 250.32 | 327.13 | 56.12 |
| 13 | 323.11 | 307.06 | 353.36 | 30.25 |
| 41 | 266.31 | 290.37 | 277.68 | 11.36 |

Table 5.5. The position, orientation, and velocities of the robot in a simple environment

| Simulation Time in Seconds | Position | | | Rotation Angle in Degree $\theta$ | Left Wheel Velocity | Right Wheel Velocity |
|---|---|---|---|---|---|---|
| | x | y | z | | | |
| **5** | 0.34 | 0.05 | 1.24 | -92.24 | 270.24 | 229.76 |
| **22** | 0.08 | 0.05 | 1.26 | -165.58 | -200.65 | 400.23 |
| **30** | 0.03 | 0.05 | 1.33 | -100.87 | 200.65 | 189.44 |
| **38** | -0.06 | 0.05 | 1.34 | -36.04 | 305.34 | -199.72 |
| **41** | -0.14 | 0.05 | 1.24 | -101.62 | 213.16 | 286.84 |

In addition, Table 5.4 shows the three ground sensors values used for the line following approach at different simulation times. It also shows the delta values which are the difference between the left and right ground sensors. Delta values are used to adjust the robot's left and right speeds to follow the line [4].

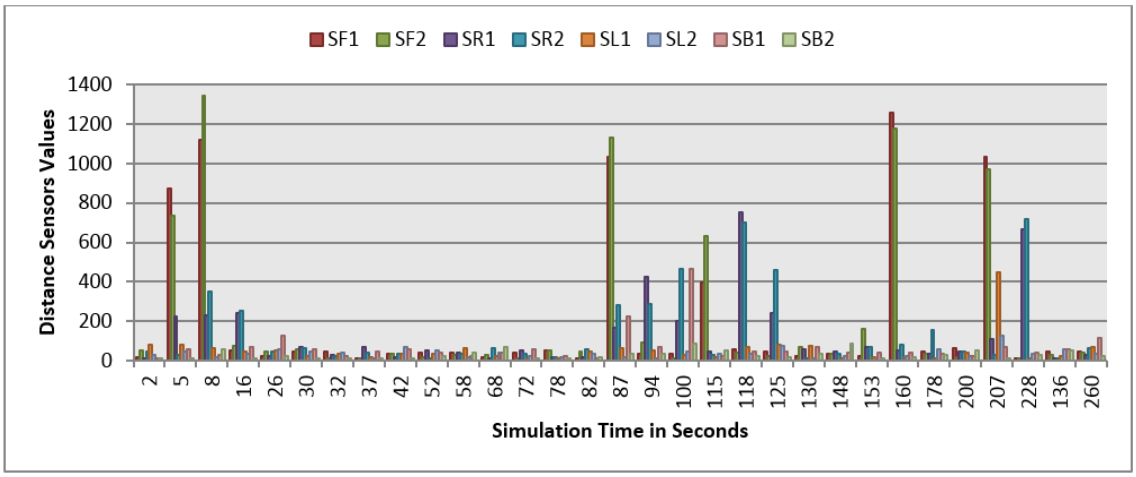
Finally, Table 5.5 shows the robot's position, orientation, and velocities. The position of the robot has been obtained through the GPS sensor. The position and orientation of the robot are obtained according to Webots Pro global coordinates system. As shown in Table 5.5, when the robot detects an obstacle at a time 22 s of the simulation, its left and right velocities are adjusted. The negative left velocity and the positive right velocity mean that the robot is turning at the left direction to avoid the obstacle. At a time of 38 s, the robot has avoided the obstacle and turned right to continue following the line [4].

### 5.4.2. Second Scenario

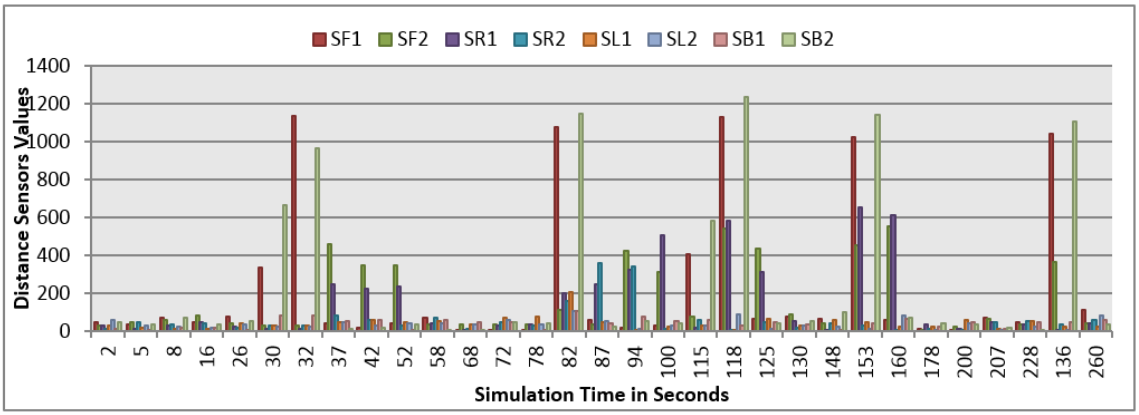
This section demonstrates the proposed model in a more complex environment where it is composed of a number of obstacles in different sizes and shapes. Two robots are running in this scenario where both are avoiding obstacles and each other as well [4]. Each robot considers the other as a dynamic obstacle. As presented in Figure 5.11, two robots (1 and 2) in opposite directions are following the line and overtaking obstacles. Figures 5.13–5.15 show all distance sensor readings, distance to obstacles in meters, and left and right velocities through the entire loop for the two robots, respectively [4].

In Figure 5.14, the distances to obstacles are obtained by the range finder camera where sometimes the obstacle is either in a distance greater than one meter or it is outside the camera field of the view. In later cases, the camera cannot measure the distances between the robots and the obstacles, which explains the gaps in Figure 5.14 a,b [4].

At the beginning of the simulation, the distance between robot 1 and the obstacle is 0.15 m as shown in Figure 5.14a. At a time of 8 seconds, robot 1 detects an obstacle where both front distance sensors (SF1 and SF2) have values greater than the threshold value as presented in Figure 5.13a. The distance between the robot and the obstacle at a time of 8 seconds has been decreased to 0.047 m as shown in Figure 5.14a. As a result, the robot will adjust its speed accordingly to avoid colliding with the obstacle.




(**a**)

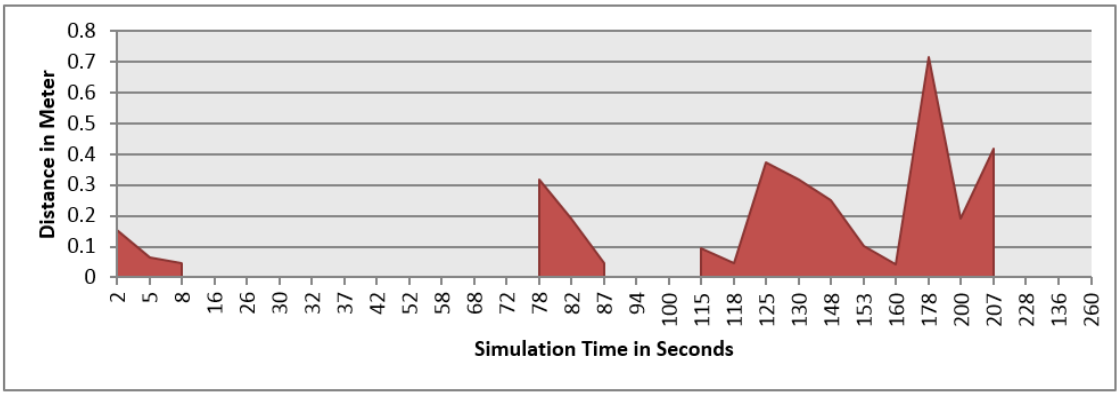



(**b**)

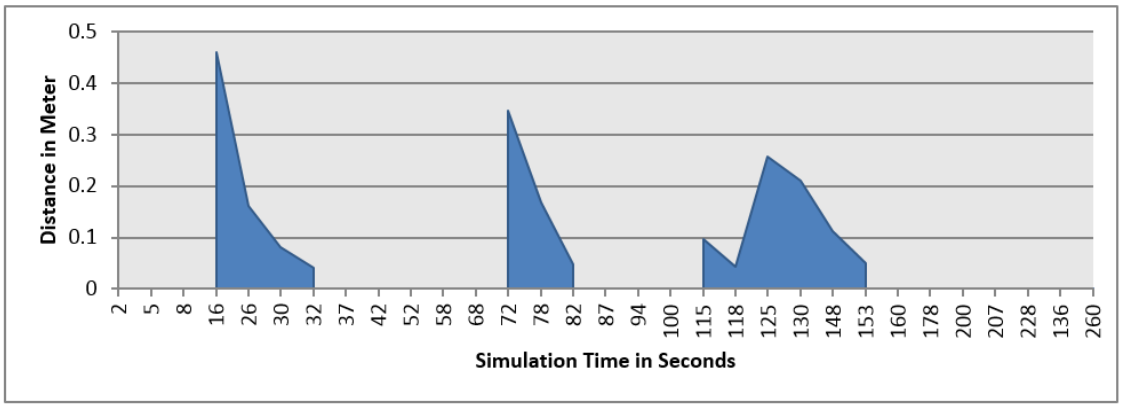
Figure 5.13**.** Distance sensor values for both robots at different simulation times.

Figure 5.15a, shows the left and right velocities for robot 1. At a time of 8 seconds, the left wheel velocity is a negative value (−168) and the right wheel velocity is a positive value (454), which indicates that robot 1 is turning left to avoid collision. At time 16 s, robot 1 is turning right around the obstacle where the left wheel velocity is 395 and the right wheel velocity is −199. After that, the robot will continue following the line until another obstacle is detected. The ground sensor values and traveled distance by left

and right wheels for both robots during the entire loop are demonstrated in Figures 5.16 and 5.17, respectively [4].



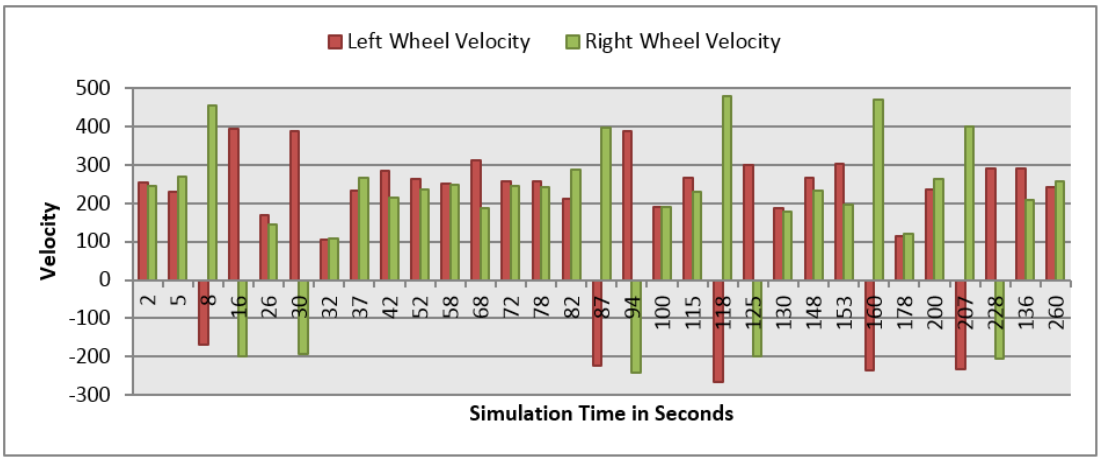

(**a**)



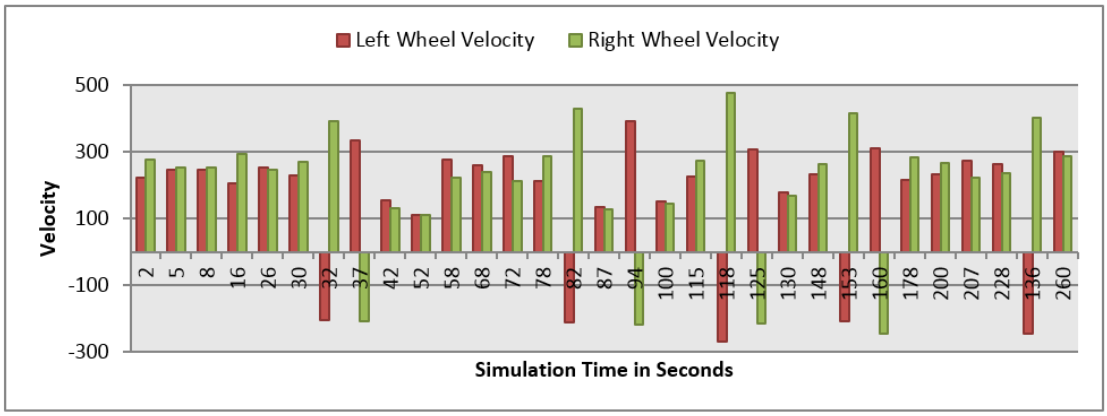

(**b**)

Figure 5.14. Distance to obstacles for both robots.

Furthermore, at time 115 s, robots 1 and 2 face each other after avoiding a couple of obstacles successfully. At that time, the distance between both robots is approximately 0.096 m. At a time of 118 seconds, the distance between them reaches 0.044 m. Both robots turn in opposite directions to avoid collision.



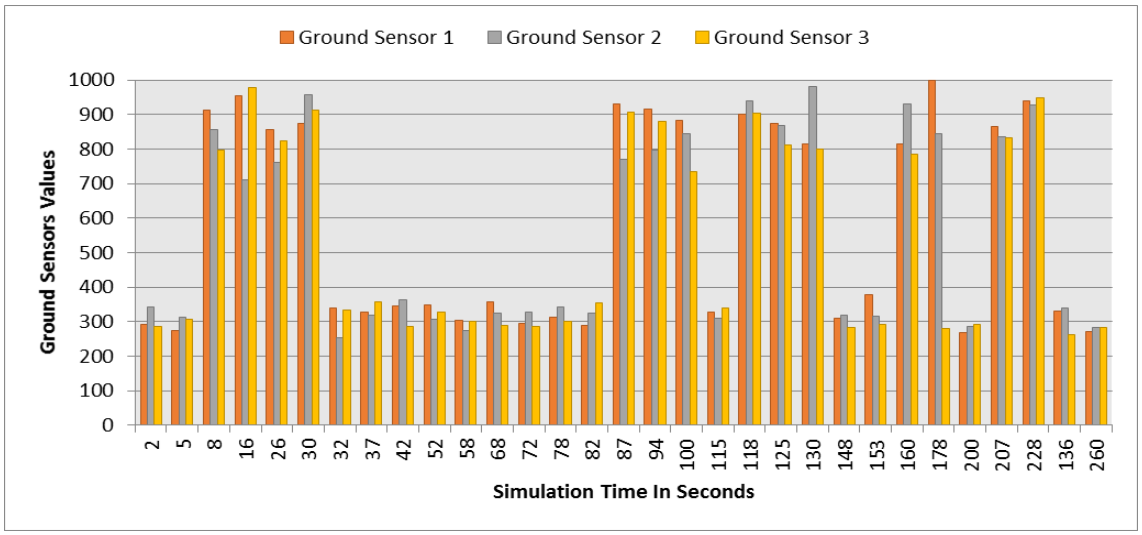

(**a**)



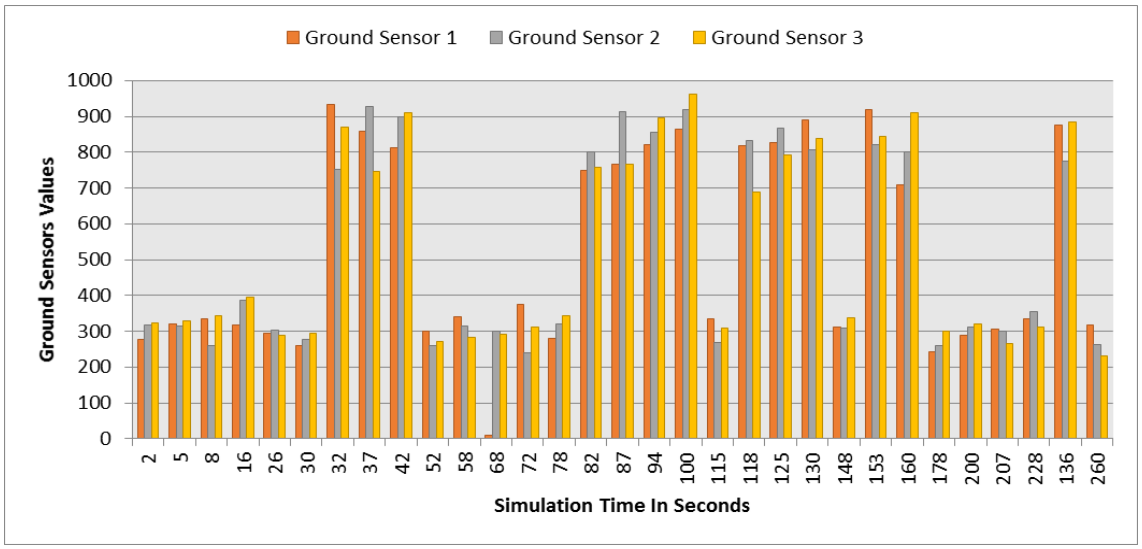

(**b**)

Figure 5.15. Left and right velocities for both robots at different simulation times.

To illustrate, at this time, the speed of robot 2 has been adjusted as shown in Figure 5.15b. Both robots will get around each other and return to follow the line. The position and orientation of both robots according to Webots Pro global coordinates system are presented in Table 5.6 [4].




(**a**)



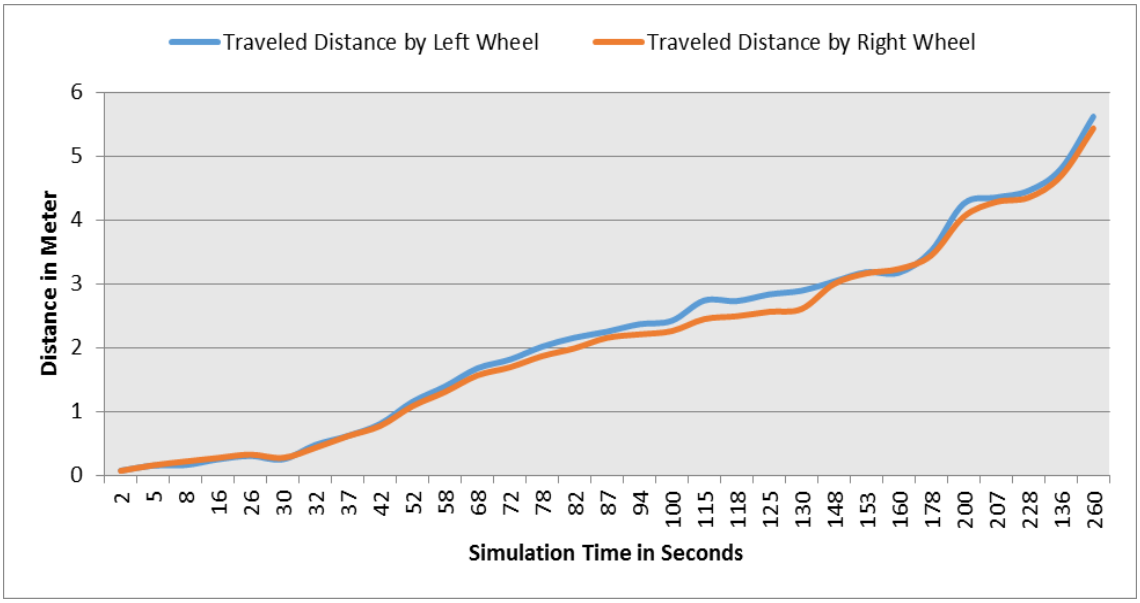

(**b**)

Figure 5.16. Ground sensors values for both robots at different simulation times.

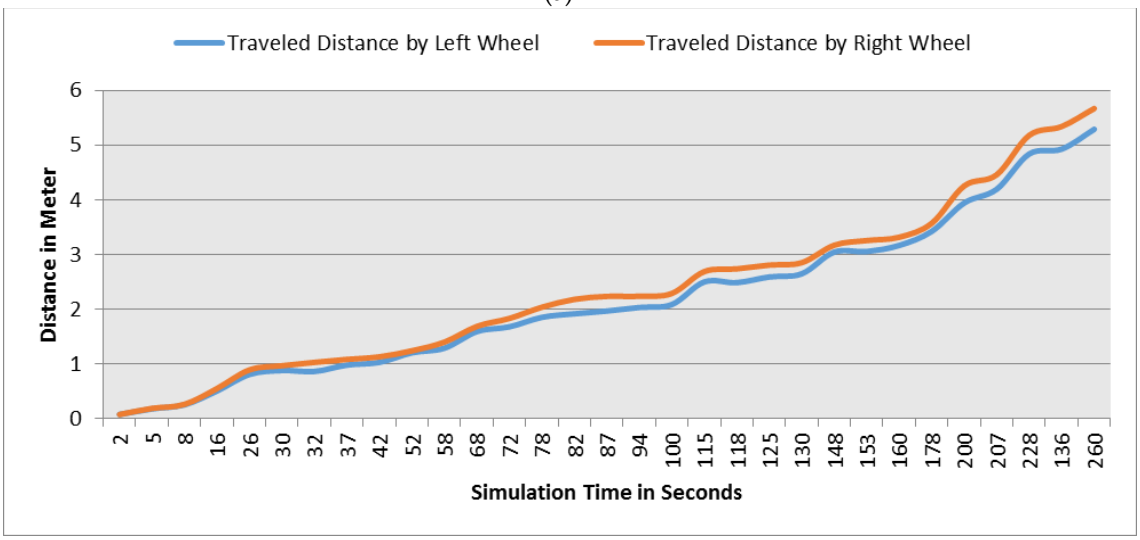
### 5.4.3 Third Scenario

In this scenario, there are more cluttered obstacles in the environment where it is more challenging for the robot to avoid them. The robot needs to adjust its speed and

orientation according to obstacles positions. Figures 5.18 and 5.19 represent the simulation of the mobile robot with many cluttered obstacles around [4].




(a)




(**b**)

Figure 5.17. Traveled distance measured by left and right wheels for both robots.

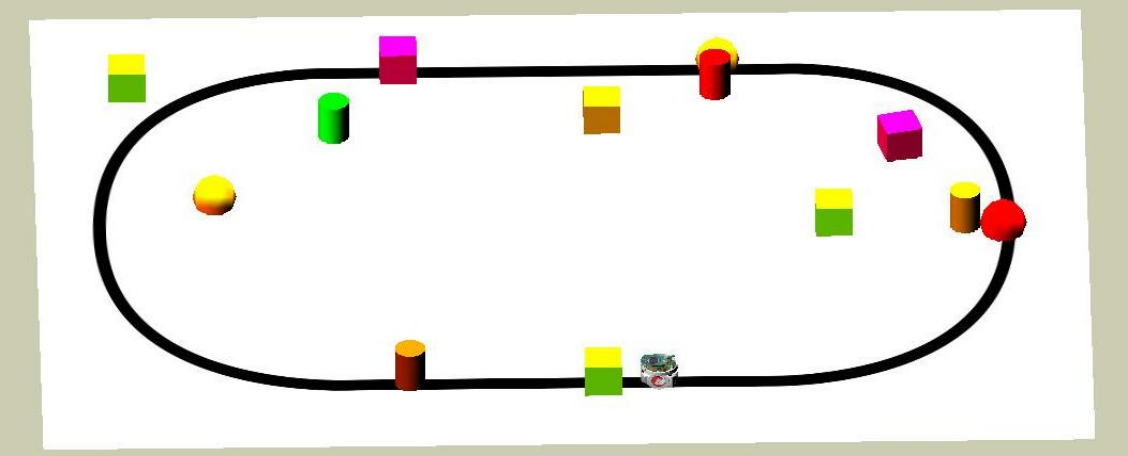


Figure 5.18. Simulation overview of the mobile robot and the environment.

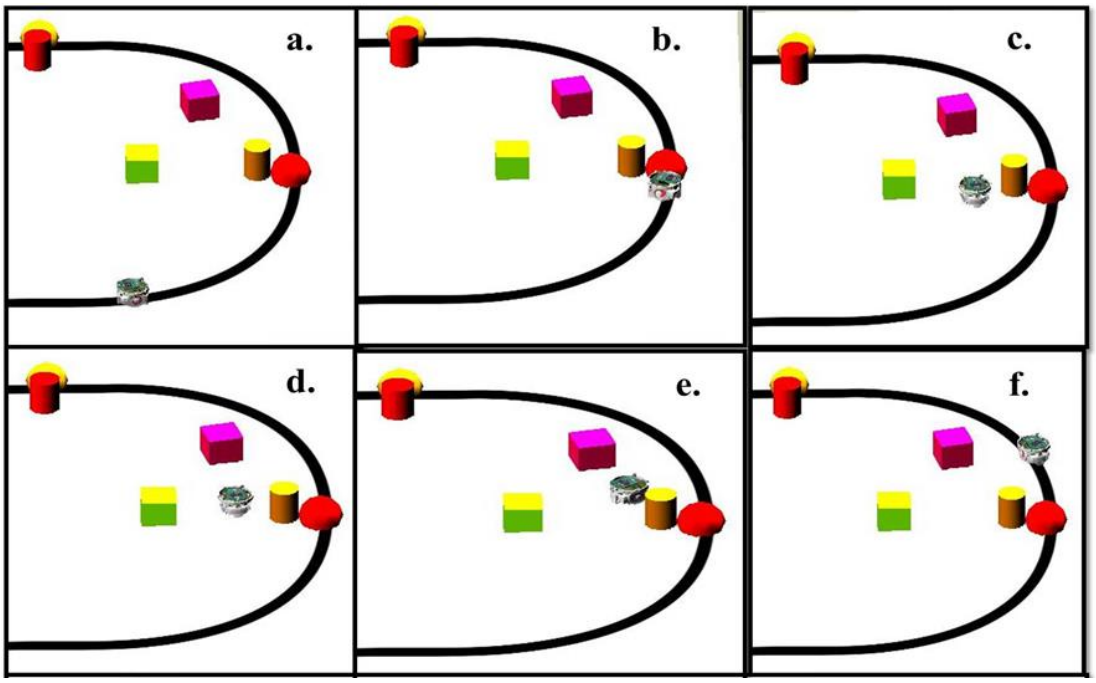



Figure 5.19. Simulation snapshots at various times.

Figure 5.20 demonstrates the distance sensor values, and Figure 5.21 shows the distance to obstacles obtained by the range finder camera at various simulation times. At the beginning of the simulation, the robot starts sensing the environment for possible obstacle detection. It also follows the predefined line using the ground sensors.

Table 5.6. The position and orientation of both robots at different simulation times.

| Simulation Time in Seconds | Robot 1 | | | | Robot 2 | | | |
|---|---|---|---|---|---|---|---|---|
| | Position | | | Rotation Angle in Degree $\theta$ | Position | | | Rotation Angle in Degree $\theta$ |
| | x | y | z | | x | y | z | |
| **2** | 0.18 | 0.05 | 0.15 | 99.39 | −0.20 | 0.05 | 0.15 | −88.26 |
| **5** | 0.27 | 0.05 | 0.16 | 48.17 | −0.31 | 0.05 | 0.15 | −93.40 |
| **8** | 0.28 | 0.05 | 0.13 | 16.01 | −0.40 | 0.05 | 0.15 | −93.94 |
| **16** | 0.33 | 0.05 | 0.07 | 80.73 | −0.62 | 0.05 | 0.23 | −125.41 |
| **26** | 0.43 | 0.05 | 0.11 | 145.54 | −0.79 | 0.05 | 0.52 | −167.63 |
| **30** | 0.46 | 0.05 | 0.15 | 145.54 | −0.79 | 0.05 | 0.59 | 110.00 |
| **32** | 0.47 | 0.05 | 0.16 | 145.54 | −0.77 | 0.05 | 0.59 | 110.00 |
| **37** | 0.59 | 0.05 | 0.17 | 107.47 | −0.72 | 0.05 | 0.62 | 174.72 |
| **42** | 0.73 | 0.05 | 0.23 | 127.69 | −0.72 | 0.05 | 0.68 | 174.71 |
| **52** | 0.90 | 0.05 | 0.51 | 167.23 | −0.78 | 0.05 | 0.76 | −120.48 |
| **58** | 0.92 | 0.05 | 0.70 | 179.96 | −0.81 | 0.05 | 0.83 | 158.72 |
| **68** | 0.87 | 0.05 | 0.99 | −154.20 | −0.71 | 0.05 | 1.10 | 142.67 |
| **72** | 0.80 | 0.05 | 1.10 | −141.02 | −0.61 | 0.05 | 1.19 | 121.69 |
| **78** | 0.65 | 0.05 | 1.21 | −111.32 | −0.44 | 0.05 | 1.25 | 95.44 |
| **82** | 0.54 | 0.05 | 1.24 | −100.42 | −0.34 | 0.05 | 1.23 | 4.69 |
| **87** | 0.41 | 0.05 | 1.26 | −169.79 | −0.34 | 0.05 | 1.17 | 4.70 |
| **94** | 0.39 | 0.05 | 1.34 | −105.05 | −0.27 | 0.05 | 1.13 | 69.45 |
| **100** | 0.31 | 0.05 | 1.36 | −105.05 | −0.21 | 0.05 | 1.14 | 134.26 |
| **115** | 0.14 | 0.05 | 1.25 | 261.32 | 0.05 | 0.05 | 1.25 | 89.07 |
| **118** | 0.13 | 0.05 | 1.28 | −161.27 | 0.07 | 0.05 | 1.22 | 11.18 |
| **125** | 0.10 | 0.05 | 1.34 | −96.53 | 0.10 | 0.05 | 1.16 | 75.91 |
| **130** | 0.04 | 0.05 | 1.34 | −96.53 | 0.15 | 0.05 | 1.14 | 75.91 |
| **148** | −0.01 | 0.05 | 1.24 | 84.02 | 0.28 | 0.05 | 1.24 | 265.34 |
| **153** | −0.18 | 0.05 | 1.24 | −101.41 | 0.36 | 0.05 | 1.22 | 4.43 |
| **160** | −0.19 | 0.05 | 1.32 | −179.93 | 0.37 | 0.05 | 1.15 | 69.20 |
| **178** | −0.43 | 0.05 | 1.28 | −56.51 | 0.46 | 0.05 | 1.25 | 84.13 |
| **200** | −0.80 | 0.05 | 0.86 | −9.46 | 0.88 | 0.05 | 0.94 | 18.56 |
| **207** | −0.83 | 0.05 | 0.75 | −81.51 | 0.91 | 0.05 | 0.78 | 6.13 |
| **228** | −0.83 | 0.05 | 0.56 | 53.63 | 0.48 | 0.05 | 0.16 | −71.84 |
| **136** | −0.74 | 0.05 | 0.37 | 28.13 | 0.39 | 0.05 | 0.20 | −170.91 |
| **260** | −0.17 | 0.05 | 0.16 | 82.96 | 0.17 | 0.05 | 0.15 | −94.52 |

As shown in Figure 5.19b, the robot turned left due to the presence of an obstacle. At 32 s of the simulation time, SF1 (front distance sensor) reached a value of 1261, which indicates that there is an obstacle detected (Figure 5.20). In addition, the range finder camera has measured the distance to that obstacle which is 0.043 meter as in Figure 5.21.

At 37 seconds, the robot detects another obstacle on its right side and moves forward as in Figure 5.19c. Then, the robot gets stuck in between two obstacles and another obstacle in front of it. As shown in Figure 5.19d, the robot tries to travel in between both obstacles. At 50 seconds, the distance between the robot and the front obstacle is 0.21 meter as in Figure 5.21. After that, the robot turns right to catch the line again as in Figure 5.19f. Figure 5.22 depicts the ground sensor values at different times, and Figure 5.23 shows the left and right wheels' velocities of the robot. At a time of 97 seconds, the robot detects another obstacle on its path and turns left as indicated in Figure 5.23. At 112 seconds, the robot detects an obstacle on its right side, which is very close to the first one and another obstacle at the front. Again, the robot tries to move in between both obstacles to recover its path. Table 5.7 summarizes the robot's position and rotation angle in degrees at various simulation times [4].

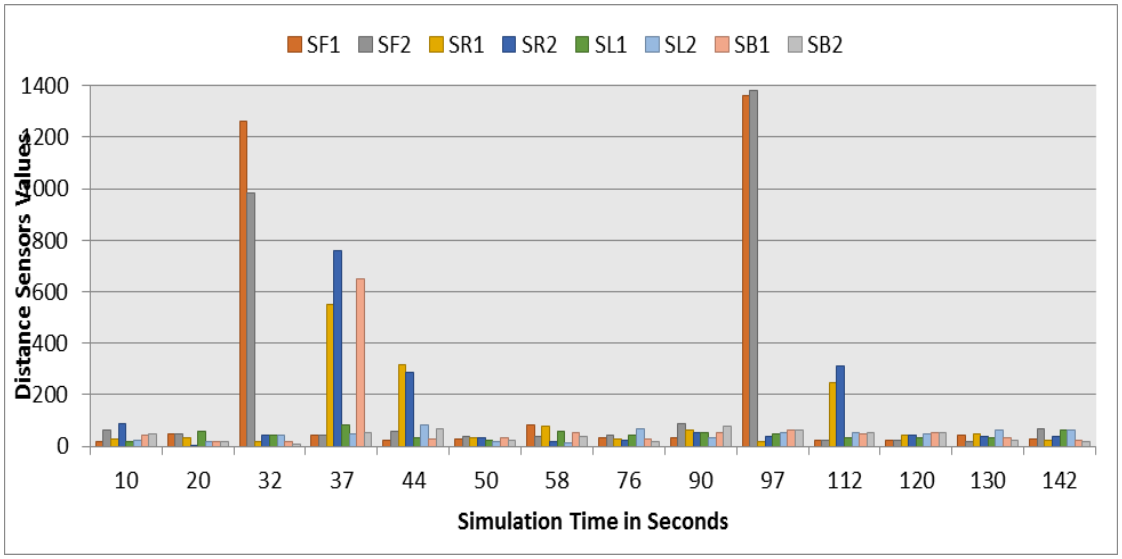



Figure 5.20. Distance sensor values at different simulation times.

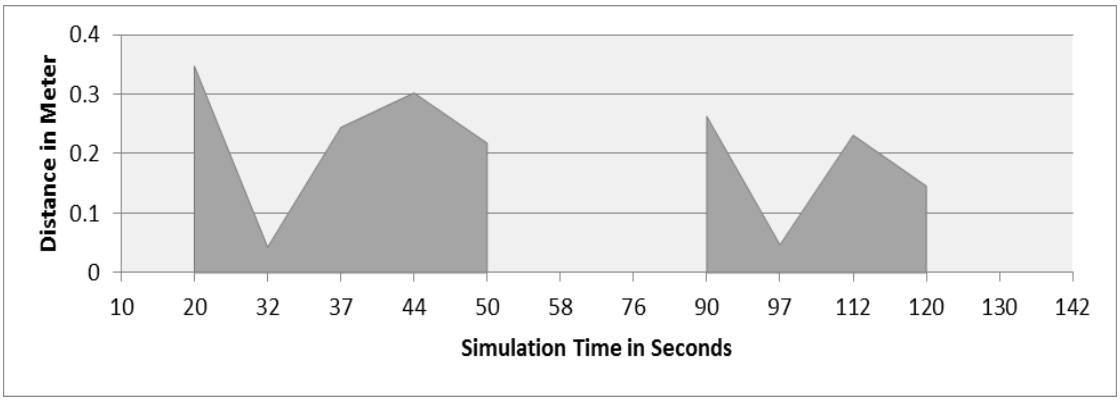



Figure 5.21. Distance to obstacles in meters at different simulation times.

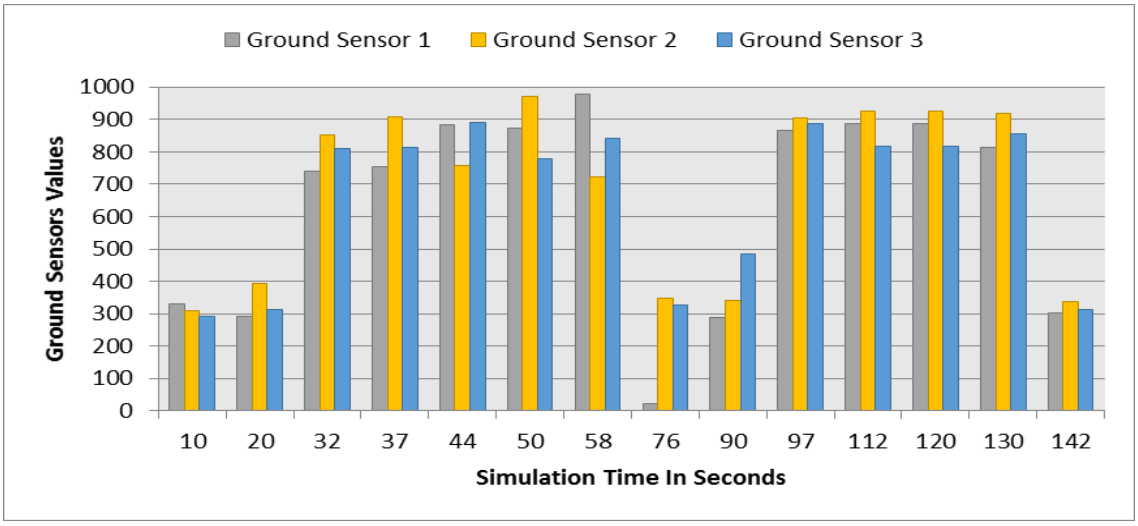



Figure 5.22. Ground sensor values at different simulation times.

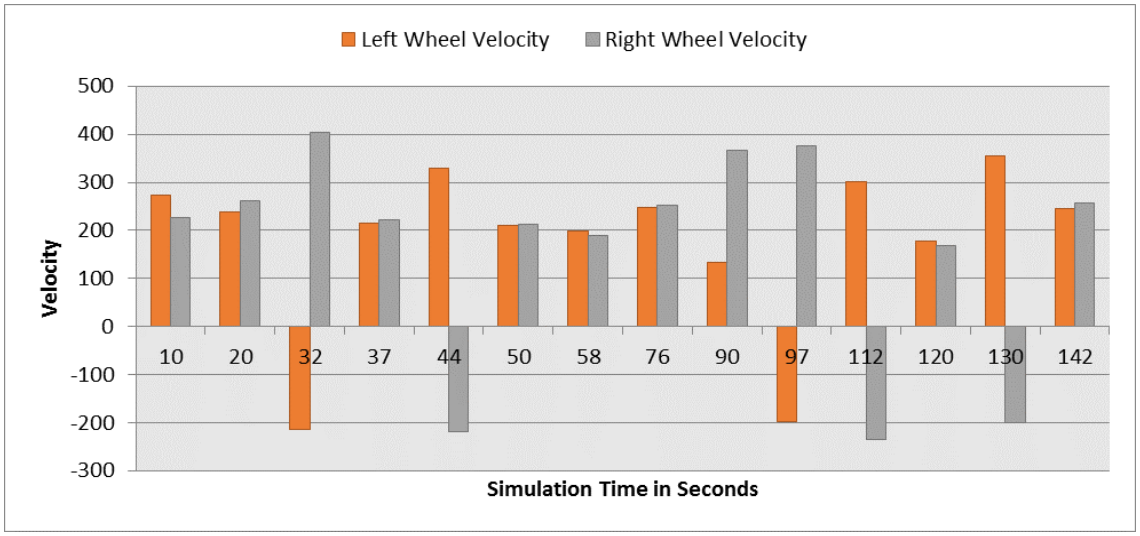



Figure 5.23. Left and right wheel velocities at different simulation times.

Table 5.7. Summaries of the robot's position and rotation angle at various simulation times.

| Simulation Time in Seconds | Position | | | Rotation Angle in Degree $\theta$ |
|---|---|---|---|---|
| | x | y | z | |
| 10s | −0.46 | 0.05 | 0.16 | −101.09 |
| 20s | −0.72 | 0.05 | 0.33 | −147.02 |
| 32s | −0.79 | 0.05 | 0.59 | 109.58 |
| 37s | −0.71 | 0.05 | 0.62 | 109.58 |
| 44s | −0.65 | 0.05 | 0.65 | 174.30 |
| 50s | −0.64 | 0.05 | 0.74 | 175.88 |
| 58s | −0.68 | 0.05 | 0.80 | −119.31 |
| 76s | −0.78 | 0.05 | 0.93 | 171.94 |
| 90s | −0.52 | 0.05 | 1.23 | 107.53 |
| 97s | −0.34 | 0.05 | 1.23 | 9.67 |
| 112s | −0.27 | 0.05 | 1.07 | 74.42 |
| 120s | −0.19 | 0.05 | 1.05 | 92.64 |
| 130s | −0.11 | 0.05 | 1.14 | 139.23 |
| 142s | 0.05 | 0.05 | 1.24 | 81.03 |

# CHAPTER 6: PERFORMANCE EVALUATION

The E-puck has successfully detected different types of obstacles (static and dynamic obstacles) with various shapes and sizes, and avoided them while it was following the line. Different scenarios have been presented with simple, complex, and challenging environments. Distance sensors and a camera are used for obstacle detection and distance measurement. The distance sensor can only detect the obstacle when the robot is very close to the obstacle, while the camera can detect it up to one meter ahead of the robot. Before applying the proposed fusion model, the distance sensors had detected the obstacle at a distance of 0.076 m from that obstacle to the robot. The camera has detected the obstacle at a distance of 0.097 m between the camera and the obstacle. On the other hand, after implementing the proposed fuzzy logic fusion methodology for collision avoidance behavior, the robot has detected the obstacle at a distance of 0.040 m. Detecting obstacles in a short distance is very efficient and beneficial, especially in a dynamic environment where the robot quickly detects obstacles that suddenly gotten in its way. Figure 6.1 demonstrates the distance to obstacle measurements by using the distance sensor, the camera, or both with the integration of fusion model. As shown in Figure 6.1, fusing both sensors outweighs the performance of using each sensor separately [4].

Furthermore, the distance traveled by the left and right robot's differential wheels is observed. As shown in Figure 6.2, the proposed fusion model has helped in reducing the distance traveled by the robot, as opposed to each sensor separately, especially at the beginning of the simulation, which saves more energy, time, and computational load [4]. Figure. 6.3 demonstrates the energy consumption level of the mobile robot with the proposed fusion system, as compared to a non-fuzzy system. At the beginning of the simulation, both cases have the same level of energy consumption. However, once the mobile robot has detected an obstacle, the energy consumed by a non-fuzzy approach has been increased whereas the proposed fusion system saves more energy. As shown in Figure 6.3, the robot detected the obstacle and tried to avoid it at 15 seconds of the simulation time. The mobile robot has consumed about 25.37 joules without the integration of the fuzzy logic technique, whereas it consumed only 18.16 joules with the fuzzy logic technique. At the time of 60 seconds in simulation, the robot has consumed less energy with the fuzzy logic technique (which is 66.24 joules) as compared to a non-fuzzy technique (which consumed 74.76 joules).

In addition, an example of the proposed model using the MATLAB rule viewer is presented in Figure 6.4. In this figure, the sensor values of SF1, SF2, SR1, and SR2 (which are the front and right distance sensors) are higher than the set threshold. As a result, there are obstacles detected at the front and right sides of the robot's position. As shown in Figure 6.4, LV has a negative value and RV has a positive value, which means that the robot turns left due to the presence of obstacles at the front and right sides [4].

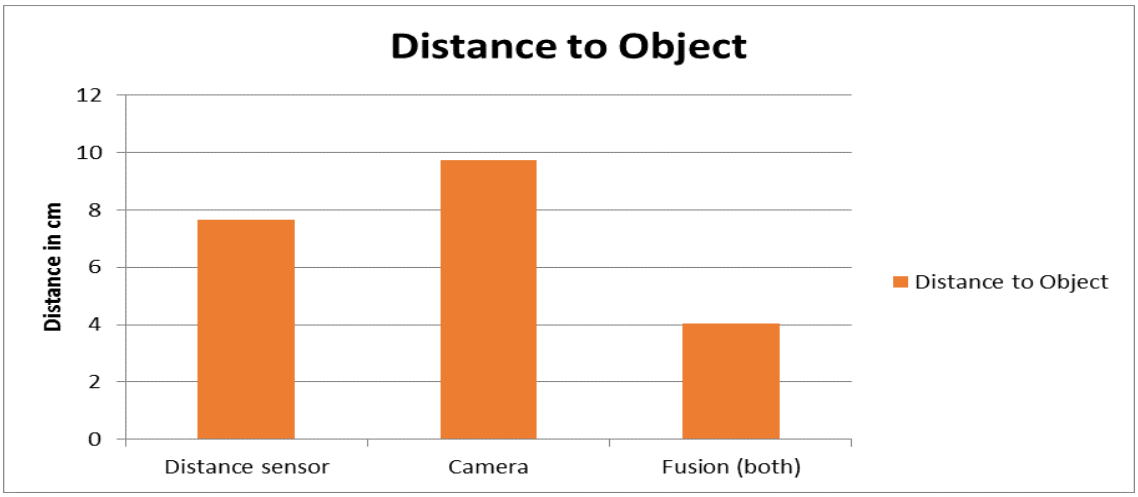



Figure 6.1. Distance measurements between the robot and the obstacle.

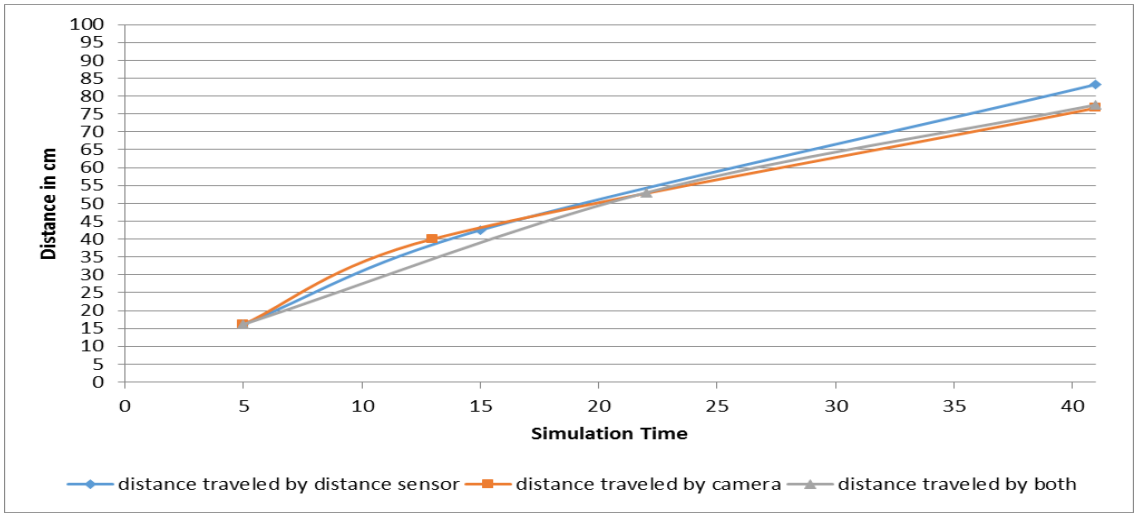



Figure 6.2. Average of distance traveled by the robot's differential wheels.

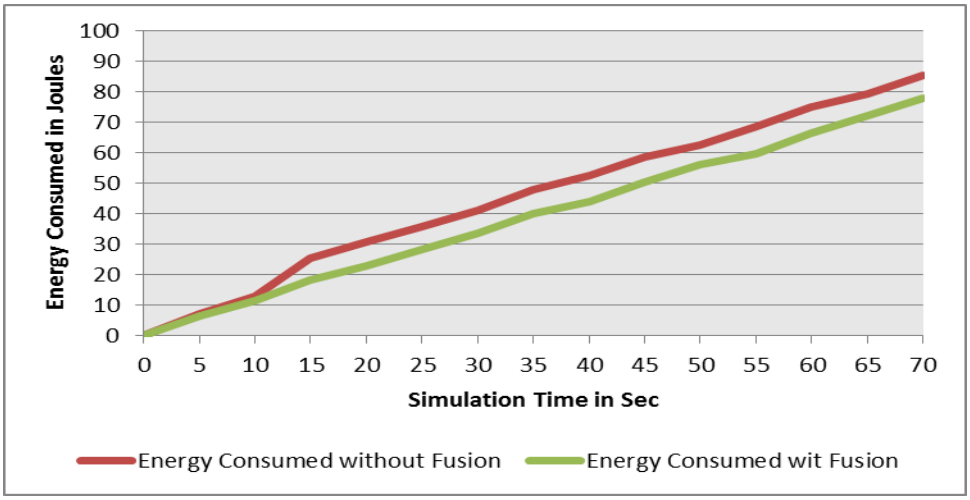



Figure 6.3. Energy consumption level

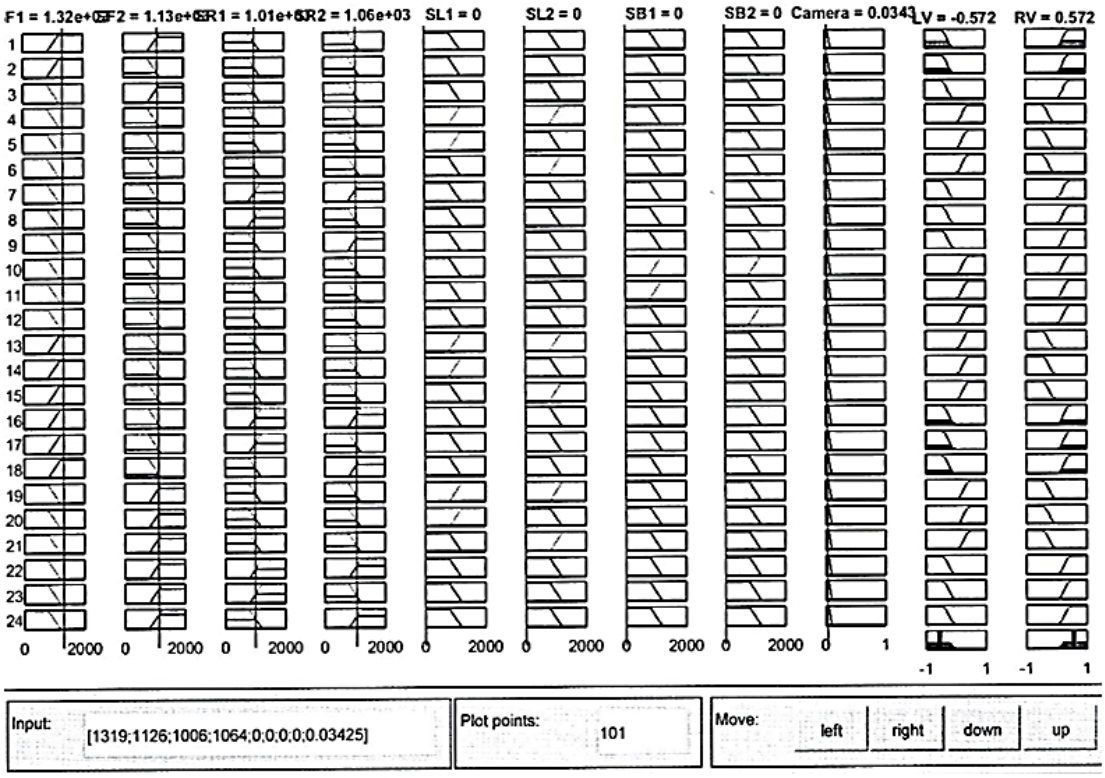



Figure 6.4**.** An example of the fusion model at MATLAB's rules viewer.

Furthermore, we tested the proposed methodology as the number of detected obstacles increased while following the line, in order to evaluate the complexity of the proposed algorithm. Figure 6.5 represents the time complexity of the proposed methodology for a one completed loop. As shown in Figure 6.5, the time taken for the mobile robot to follow the line without any obstacles detection due to the absence of obstacles in the environment is 157 seconds. If one obstacle is detected, the robot will take 190 seconds to follow the line while avoiding the obstacle. As shown in Figure 6.5, the time taken for avoiding obstacles and following the path increases proportionally to the number of detected obstacles in term of *O(n log n).*

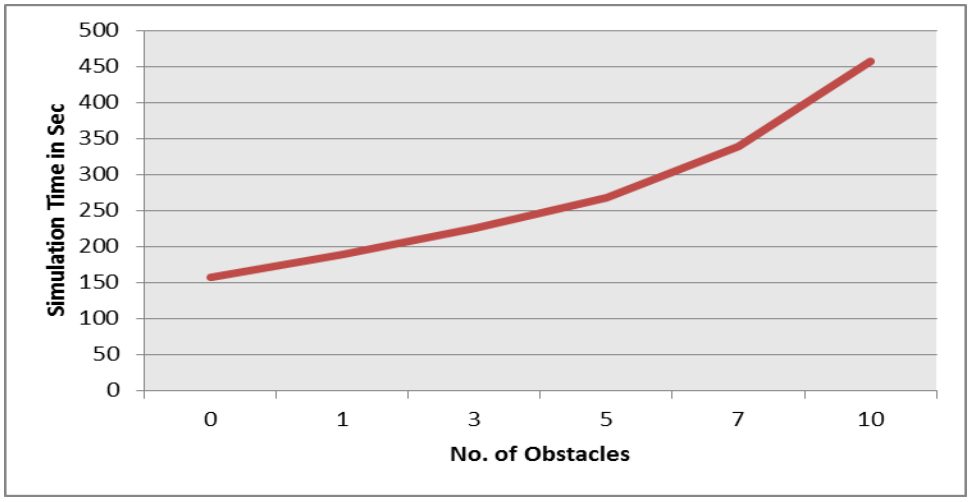



Figure 6.5. Time complexity of the proposed methodology

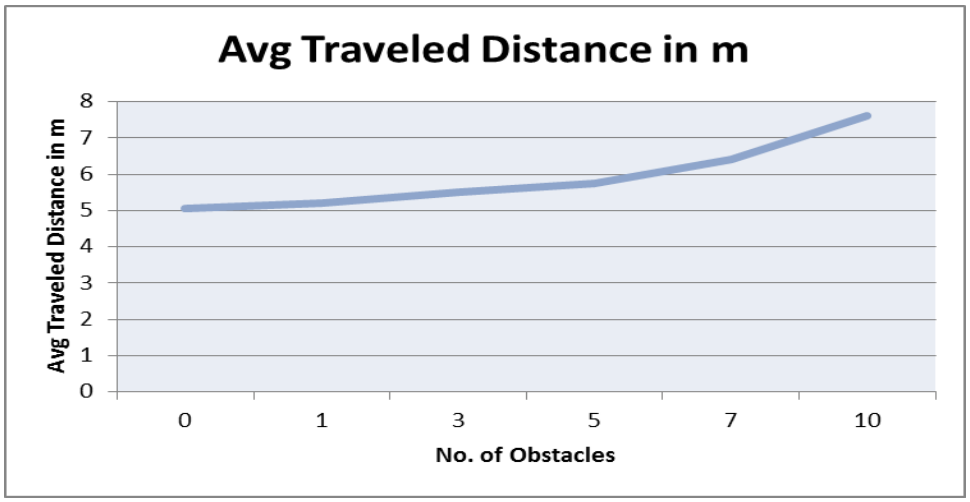



Figure 6.6. Average traveled distance by the mobile robot per number of obstacles.

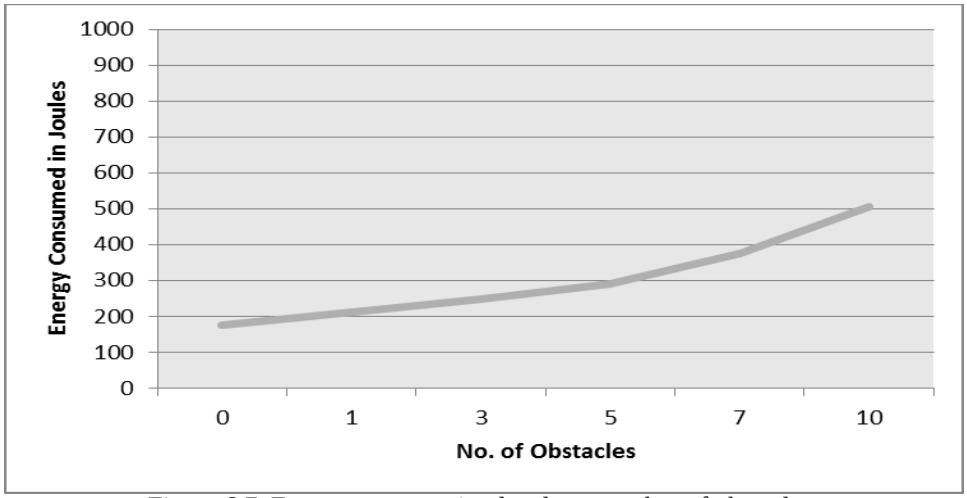



Figure 6.7. Energy consumption level per number of obstacles.

Moreover, Figures 6.6 and 6.7 show the average traveled distance in m and the energy consumption by the mobile robot in one completed loop as the number of detected obstacles increases, respectively. In the case no obstacles are detected, the mobile robot has traveled about 5 meters and consumed 174.72 joules. If one obstacle is detected, the average traveled distance by the robot is 5.22 m, whereas the energy consumed by the robot is 211.08 joules. In case 7 obstacles are detected, the robot has traveled 6.4 m and consumed 376.27 joules. As a result, the average traveled distance and the energy consumption level of the mobile robot grow proportionally to the number of detected obstacles.

Moreover, the computational complexity of the fuzzy logic controller can be calculated based on the number of operations needed for each process of the fuzzy system (such as fuzzification, inference, and defuzzification). The number of operations for each part is affected by various parameters: the number of inputs and outputs, the number of the fuzzy sets of the inputs and outputs, the number of rules, etc [96]. To illustrate, the number of operations required for the fuzzification process of the fuzzy logic controller, where nonspecific fuzzy sets are used is as follows [96]:

$$(70X_{IF}+29X_{IF}\,X_{ID}+8)*\,I \tag{6.1}$$

Where $X_{IF}$ *is* the number of the input fuzzy sets, and $I$ is the number of inputs.

In addition, the number of operations required for the defuzzification process of the fuzzy logic controller, where center of gravity method is used is as follows [96]:

$$(39X_{OD}+5)*R+15 \tag{6.2}$$

Where *R* is the number of defined rules used in the fuzzy inference system.

In [97], two fuzzy logic controllers were designed. One for the line following approach, the other is for the obstacle detection along the path. The inputs are obtained from the IR and proximity sensors and the output is the desired speed of two wheels of the mobile robot. The technique was implemented on a mobile robot with a micro-controller. In [98], fuzzy logic controller was used for the obstacle avoidance approach for safe path planning, where the inputs are eight IR sensors and the output is the motor speed of the mobile robot. The number of fuzzy logic rules applied is 256; this technique was implemented on the E-puck robot.

The total number of operations for each part of the fuzzy logic system has been calculated for [97], [98], and the proposed technique, in order to compare and evaluate the proposed technique versus others that use the fuzzy logic system for mobile robot navigation. Figure 6.8 shows the total number of operations for each technique. As shown in Figure 6.8, the technique in [98] has the highest number of operations, due to a high number of rules used (which is 256 rules). In addition, the proposed technique has a lower total number of operations compared to the technique in [97], due to the fact that the later technique had two fuzzy logic controllers for the mobile robot navigation (which led to a higher number of operations required).

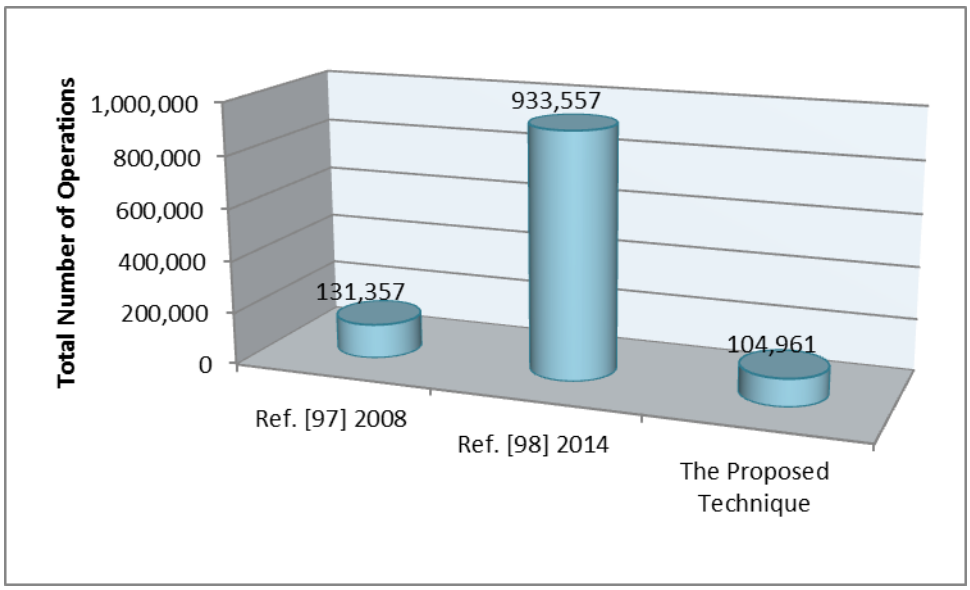



Figure 6.8. The comparison of the total number of operations for each technique.

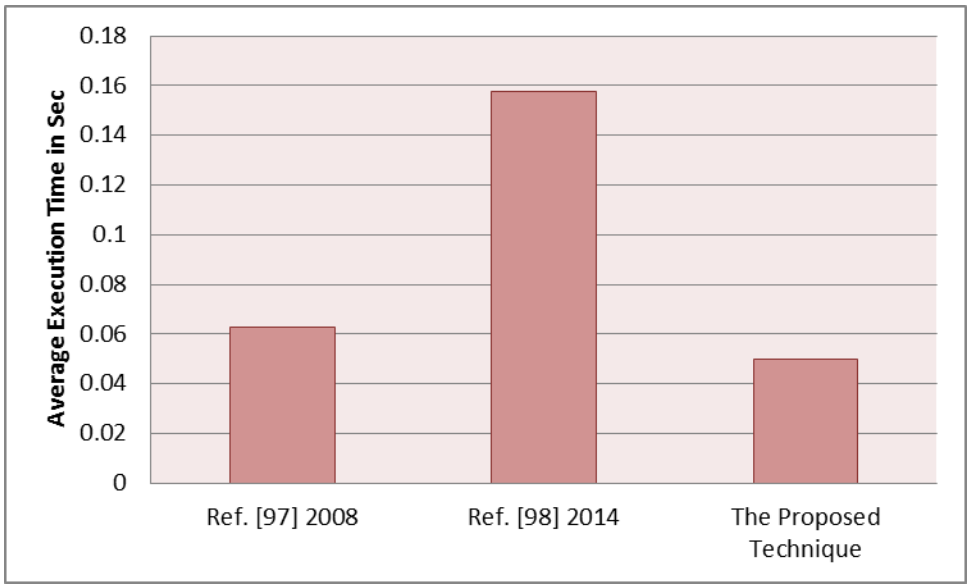



Figure 6.9. The comparison of the average execution time for each technique.

Besides the total number of operations, the average execution time was measured for each technique. The execution time refers to the CPU time needed to implement the fuzzy inference system in MATLAB. The average execution time is computed for ten different measurements for each technique. Figure 6.9 represents the average execution

time in seconds. As shown in Figure 6.9, the technique in [97] required 0.0628 seconds, the technique in [98] required 0.1576 seconds, and the proposed technique required 0.0499 seconds. The techniques in [97] and [98] needed higher average execution times due to a higher number of operations.

Furthermore, the energy consumption was measured for each technique. Figure 6.10 indicates the level of energy consumed. As shown in Figure 6.10, the proposed technique consumed less energy (which is 211.08 joules) compared to the other techniques (262.26 joules in [97], and 334.15 joules in [98]). The energy consumption can be affected by different factors such as the total number of operations, the average execution times, and the traveled distance. The proposed technique shows lower total number of operations, average execution time, energy consumption, and less distance traveled by the mobile robot wheels. The comparison between the proposed technique and the techniques in [97] and [98] is summarized in Table 6.1.

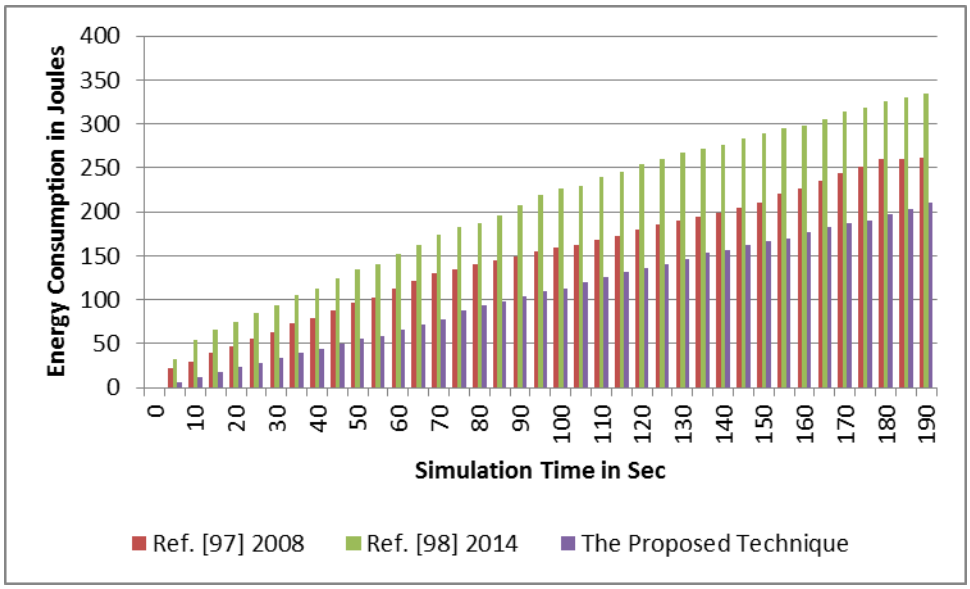



Figure 6.10. The comparison of the level of energy consumption for each technique.

Our approach aims at following the robot along a predefined path (black line on a white surface), while avoiding multiple obstacles on its way such as static, dynamic, and cluttered obstacles. Applying the fuzzy logic fusion has successfully reduced the distance traveled by the robot's wheels, as well as minimized the distance between the robot and the obstacle detected as compared to a non-fuzzy logic approach; this is beneficial in a dynamic environment [4]. The proposed fusion model also leads to lower energy consumption by the mobile robot due to less traveled time, as compared to the non-fusion technique.

In addition, the proposed technique has a lower total number of operations, average execution time, energy consumption, and less distance traveled compared to the techniques in [97] and [98]; this validates the efficiency and the performance of the proposed fuzzy logic technique.

Table 6.1. The comparison between the proposed technique and the other techniques.

| **Comparison** | **Ref. [97]**<br>**2008** | **Ref. [98]**<br>**2014** | **The Proposed Technique** |
|---|---|---|---|
| **Total Number of Operations** | 131,357 | 933,557 | 104,961 |
| **Average Execution Time in Sec** | 0.0628 | 0.1576 | 0.0499 |

| | | | |
|---|---|---|---|
| **Energy Consumption in Joules** | 262.26 | 334.15 | 211.08 |
| **Average Traveled Distance in m** | 5.39 | 5.68 | 5.22 |
| **Remarks** | 2 fuzzy logic controllers used, one for line following and one for collision avoidance | Used 256 rules | Low number of rules used (only 24 rules) |

# CHAPTER 7: CONCLUSIONS AND FUTURE WORK

Recently, mobile robot navigation in an unknown environment has been the research focus in the mobile robot intelligence control domain. We developed a line follower robot that has the ability to detect and avoid any obstacles that emerge on its path. It depends on the use of infrared sensors (distance sensors and ground sensors) that are used to measure and obtain the distance and orientation of the robot. Furthermore, a multisensory fusion-based model was proposed for the collision avoidance and path follower mobile robot. Eight distance sensors and a range finder camera were used for the collision avoidance behavior, where three ground sensors were used for the line following approach. In addition, a GPS was used to obtain the robot's position. The fusion model proposed is based on the fuzzy logic system, which is composed of nine inputs, two outputs, and twenty four fuzzy rules. Multiple membership functions for inputs and outputs are developed.

The data is utilized in the Webots Pro simulator to validate the effectiveness and the performance of the proposed technique. In addition, the proposed fusion model of the mobile robot has been successfully tested in simulation and real-time experiments. Different scenarios have been presented with simple, complex, and challenging

environments. The robot detected static and dynamic obstacles with different shapes and sizes in a short distance range, which is very efficient in dynamic environment. The distance traveled by the robot was reduced using the fusion model, which reduces energy and computational consumptions, and time.

Future work will concentrate on integrating other learning strategies (such as Artificial Neural Networks and Adaptive Neuro Fuzzy Inference System (ANFIS) along with the proposed methodology for the collision avoidance and path following mobile robot. These learning strategies help in faster adaption and learning of the environment, which are suitable for more complex scenarios. The Neuro-Fuzzy controller can be applied in various scenarios, which include more complicated paths and terrain to evaluate its effectiveness (especially in real-time applications). In addition, using genetic algorithms will optimize the membership functions of inputs and output, as well as the rules.

Furthermore, the proposed methodology can be extended to include two separate controllers: one for the collision avoidance approach, the other for the line following approach. Adding other types of sensors such as laser or ultrasonic sensors for the collision avoidance technique and the camera as an example for the line following technique will optimize the results for higher accuracy and robustness.

# REFERENCES

[1] L. E. Zarate, M. Becker, B. D. M. Garrido, and H. S. C. Rocha, “An artificial neural network structure able to obstacle avoidance behavior used in mobile robots,” *IEEE 28th Annual Conference of the Industrial Electronics Society*, pp. 2457-246, 2002.

[2] S. X. Yang, and C. Luo, “A neural network approach to complete coverage path planning,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 1, no. 34, pp. 718-724, 2004.

[3] M. M. Almasri, A. M. Alajlan and K. M. Elleithy, "Trajectory Planning and Collision Avoidance Algorithm for Mobile Robotics System," in *IEEE Sensors Journal*, vol. 16, no. 12, pp. 5021-5028, June15, 2016.

[4] M. Almasri, K. Elleithy, and A. Alajlan, “Sensor Fusion Based Model for Collision Free Mobile Robot Navigation,” *Sensors*, vol. 16, no. 1, p. 24, Dec. 2015.

[5] D. Floreano, J. Godjevac, A. Martinoli, F. Mondada, and J. D. Nicoud, “Design, control, and applications of autonomous mobile robots,” in *Advances in Intelligent Autonomous Systems*, Springer Netherlands, pp. 159-186, 1999.

[6] L. Zeng, and G. M, “Mobile Robot Collision Avoidance in Human Environments,” *International Journal of Advanced Robotic Systems*, vol.10, no. 41, 2013.

[7] A. Abdelgawad and M. Bayoumi, *Resource-Aware Data Fusion Algorithms for Wireless Sensor Networks*, Boston, MA: Springer US, 2012.

[8] S. V and C. Chandraseka, "Energy Efficient Multipath Data Fusion Technique for Wireless Sensor Networks," *ACEEE International Journal on Network Security*, vol. 3, no. 8, 2012.

[9] R. Brooks and S. Iyengar, *Multi-Sensor Fusion: Fundamentals and Applications with Software*, Upper Saddle River, NJ: Prentice-Hall, Inc., 1998.

[10] C. Siaterlis and B. Maglaris, "Towards multisensor data fusion for DoS detection," In *Proceedings of the 2004 ACM Symposium on Applied Computing, ACM Press*, Nicosia, Cyprus, pp. 439–446, 2004.

[11] T. Bass, "Intrusion detection systems and multisensor data fusion," *Commun. ACM 2000*, vol. 43, no. 2, pp. 99-105, 2000.

[12] M. M. Almasri and K. M. Elleithy, "Data Fusion in WSNs: Architecture, Taxonomy, Evaluation of Techniques, and Challenges," *International Journal of Scientific & Engineering Research*, vol. 6, no. 4, pp. 1620 – 1636, 2015.

[13] E. Nakamura, A. Loureiro, and A. Frery, "Information fusion for wireless sensor networks: methods, models, and classifications," *ACM Comput. Surv.*, vol. 39, no. 3, 2007.

[14] V. Borges, and W. Jeberson, "Survey of Context Information Fusion for Sensor Networks based Ubiquitous Systems," In *J. Sens. Actuator Netw*, 2013.

[15] T. Bayes, and R. Price, "An essay towards solving a problem in the doctrine of chances," *Philosophical Transactions of the Royal Society*, vol. 53 , pp. 370–418, 1763.

[16] H. Pan, Z. Liang, T. Anastasio, and T. Huang, "A hybrid NN-Bayesian architecture for information fusion," in *Proceedings of the 1998 International Conference on Image Processing (ICIP'98)*, Chicago, IL. 1, pp. 368–371, 1998.

[17] C. Cou´E, T. Fraichard, P. Bessiere, and E. Mazer, "Multi-sensor data fusion using Bayesian programming: An automotive application," in *IEEE/RSJ International Conference on Intelligent Robots and System*, Lausanne, Switzerland, pp. 141–146, 2002.

[18] A. Dempster, "A generalization of Bayesian inference," J. Royal Stat. Soc., Series B, pp. 205–247, 1968.

[19] G. Shafer, *A mathematical theory of evidence*, Princeton, NJ: Princeton University Press, 1976.

[20] G. Provan, "A logic-based analysis of Dempster-Shafer theory," *International Journal of Approximate Reasoning*, vol. 4, no. 5, pp. 451–495, 1990.

[21] T. Garvey, J. Lowrance, and M. Fischler, "An inference technique for integrating knowledge from disparate sources," in *Proceedings of the 7th international joint conference on Artificial intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 319-325, 1981.

[22] D. Friedlander and S. Phoha, "Semantic information fusion for coordinated signal processing in mobile sensor networks," *Int. J. High Perf. Comput*. 16, pp. 235–241, 2002.

[23] D. Friedlander, "Semantic information extraction," in *Distributed Sensor Networks*; S. S. Iyengar, S.S.; Brooks, R.R., Eds., CRC Press, Boca Raton, pp. 409–417, 2005.

[24] V. Novak, I. Perfilieva, J. Mockor, and J. Moickoir, *Mathematical principles of fuzzy logic*, Vol. 517. Boston, MA: Kluwer Academic Publishers, 1999.

[25] L. Zadeh, "Fuzzy sets, fuzzy logic, and fuzzy systems", selected papers by Lotfi A. Zadeh. NJ: World Scientific Publishing Co., Inc, pp. 94–102, 1996.

[26] J. Jang, C. Sun, and E. Mizutani, *A Computational Approach to Learning and Machine Intelligence*, New Jersey: Prentice Hall, 1997.

[27] W. Su and T. Bougiouklis, "Modeling of data fusion algorithms in cluster-based Wireless Sensor Networks," *Signals, Systems and Computers*, 42nd Asilomar Conference on, Pacific Grove, CA, pp. 868-872, 2008.

[28] W. Su, and T. Bougiouklis, "Data fusion algorithms in cluster-based wireless sensor networks using fuzzy logic theory," in *Proceedings of the 11th Conference on 11th WSEAS International Conference on Communications (ICCOM'07)*, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2007; Mastorakis, N.E.; Kartalopoulos, S.; Simian, D.; Varonides, A.; Mladenov, V.; Bojkovic, Z.; Antonidakis, E., Eds., pp. 291-299, 2007.

[29] W. Chan Yet and U. Qidwai, "Intelligent sensor network for obstacle avoidance strategy," in *Proceedings of IEEE Conference on Sensors*, pp. 405–408, 2005.

[30] M. Yusuf and T. Haider, "Energy-aware fuzzy routing for wireless sensor networks," in *IEEE International Conference on Emerging Technologies (ICET'05). IEEE*, Islamiabad, Pakistan, pp. 63–69, 2005.

[31] L. Zadeh, " Fuzzy logic: computing with words," *Fuzzy Systems, IEEE Transactions*., vol. 4, no. 2, pp. 103–111, 1996.

[32] P. Bonissone, "Soft computing: The convergence of emerging reasoning technologies," *Soft Comput*, vol. 1, no. 1, pp. 6-18, 1997.

[33] M. Roth, "Survey of neural network technology for automatic target recognition," *Trans. Neur*. Netw, vol. 1, no. 1, pp. 28-43, 1990.

[34] T. Lewis and D. Powers, "Audio-visual speech recognition using red exclusion and neural networks," in *Proceedings of the twenty-fifth Australasian conference on Computer science*, Melbourne, Victoria, Australia, pp. 149-156, 2002.

[35] C. Peirce, "Abduction and induction," in *Philosophical Writings of Peirce*, Peirce, C. S.; Buchler, J., Eds.; Dover, New York, pp. 150–156, 1955.

[36] L. de Campos, J. Gamez, and S. Moral, "Partial abductive inference in Bayesian belief networks – an evolutionary computation approach by using problem-specific genetic operators," *IEEE Transactions on Evolutionary Computation 2002*, vol. 6, no. 2, pp. 105-131, 2002.

[37] R. Mooney, "Integrating abduction and induction in machine learning," *in Abduction and Induction, Essays on their Relation and Integration*; Flach, P.A.; Kakas, A.C., Eds.; Applied Logic Series. Kluwer: New York, 336p, 2000.

[38] J. Aguero and A. Vargas, “Inference of operative configuration of distribution networks using fuzzy logic techniques—part II: Extended real-time model,” *IEEE Trans. Power Syst*, vol. 20, no. 3, pp. 1562–1569, 2005.

[39] B. Bracio, W. Hom, and D. Moller, “Sensor fusion in biomedical systems,” in *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, Chicago, IL, vol. 3, pp. 1387–1390, 1997.

[40] C. Brown, H. Durrant-Whyte, J. Leonard, B. Rao, and B. Steer, “Distributed data fusion using Kalman filtering: A robotics application”, in *Data Fusion in Robotics and Machine Intelligence*, San Diego, CA, Abidi, M.A.; Gonzalez, R.C. Eds, pp. 267–309, 1992.

[41] T. Schmitt, R. Hanek, M. Beetz, S. Buck, and B. Radig, “Cooperative probabilistic state estimation for vision-based autonomous mobile robots,” *IEEE Trans. Robotics Autom*., vol. 18, no. 5, pp. 670–684, 2002.

[42] Y. Yuan and M. Kam, “Distributed decision fusion with a random-access channel for sensor network applications,” *IEEE Trans. Instr. Meas*., vol. 53, no. 4, pp. 1339-1344, 2004.

[43] D. Crisan and A. Doucet, “A survey of convergence results on particle filtering methods for practitioners,” *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736-746, 2002.

[44] P. Nordlund, F. Gunnarsson, and F. Gustafsson, “Particle filters for positioning in wireless networks,” in *Proceedings of the XI European Signal Processing Conference (EURSIPCO'02).* TeSA, Toulouse, France, pp. 311-314, 2002.

[45] M. Bolic, “Architectures for Efficient Implementation of Particle Filters,” Ph.D. Dissertation, State University of New York at Stony Brook, Stony Brook, NY, USA, Advisor(s) Petar M. Djuric. AAI3149104, 2004.

[46] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, “Distributed regression: an efficient framework for modeling sensor network data,” in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. Berkeley, California, USA, 2004.

[47] R. Kalman, “A new approach to linear filtering and prediction problems,” *Trans. ASME J. Basic Engin*, vol. 82, no. 1, pp. 35-45, 1960.

[48] R. Luo and M. Kay, “Data fusion and sensor integration: State-of-the-art 1990s,” in *Data Fusion in Robotics and Machine Intelligence*, Abidi, M.A.; Gonzalez, R.C., Eds., Academic Press, Inc., San Diego, CA, pp. 7-135, 1992.

[49] G. Welch and G Bishop, “An Introduction to the Kalman Filter,” University of North Carolina at Chapel Hill, Chapel Hill, NC, 1995.

[50] S. Julier and J. Uhlmann, “A new extension of the Kalman filter to nonlinear systems,” in *Signal Processing, Sensor Fusion, and Target Recognition VI*. SPIE, San Diego, pp. 182—193, 1997.

[51] L. Xiao, S. Boyd, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” In *Proceedings of the 4th international symposium on Information processing in sensor networks*, Los Angeles, California, 2005.

[52] L. Xiao, S. Boyd, and S. Lall, “A space-time diffusion scheme for peer-to-peer least-squares estimation,” In *Proceedings of the 5th international conference on Information processing in sensor networks*, Nashville, Tennessee, USA, 2006.

[53] S. Smith, *The scientist and engineer's guide to digital signal processing*, San Diego, CA: California Technical Publishing, 1997.

[54] A. Hoang and M. Motani, “Collaborative Broadcasting And Compression In Cluster-Based Wireless Sensor networks,” in *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN'05). IEEE*. Istanbul, Turkey, pp. 197-206, 2005.

[55] Z. Xiong, A. Liveris, and S. Cheng,” Distributed source coding for sensor networks,” *IEEE Sig. Proc. Mag*, vol. 21, no. 5, pp. 80-94, 2004.

[56] E. Nakamura and A. Loureiro, “Information fusion in wireless sensor networks,” in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08). ACM*, New York, NY, USA, pp. 1365-1372, 2008.

[57] S. Pradhan and K. Ramchandran, “Distributed source coding using syndromes (DISCUS): design and construction,” *IEEE Transactions on Information Theory*, vol. 49, no. 3, pp. 626-643, 2003.

[58] D. Petrovic, R. Shah, K. Ramchandran, and J. Rabaey, "Data funneling: routing with aggregation and compression for wireless sensor networks," in *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE 2003 IEEE International Workshop on*, pp. 156–162, 2003.

[59] J. Kulik, W. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks*," Wireless Networks*, vol. 8, no. 2-3, pp. 169-185, 2002.

[60] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. Boston, Massachusetts, USA, pp. 56-67, 2000.

[61] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 2-16, 2003.

[62] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on In-formation processing in sensor networks*, Berkeley, California, USA, 2004.

[63] V. Gupta, and R. Pandey, "Data fusion and topology control in wireless sensor networks," *WSEAS Trans. Sig. Proc*, vol. 4, no. 4, pp. 150-172, 2008.

[64] P. Varshney, *Distributed Detection and Data Fusion*, Secaucus, NJ: Springer-Verlag New York, Inc., 1996.

[65] M. Ahmed and G. Pottie, "Fusion in the context of information theory," in *Distributed Sensor Networks,* Iyengar, S.S.; Brooks, R.R., Eds.; CRC Press: Boca Raton, pp. 419-436, 2005.

[66] K. Marzullo, "Tolerating failures of continuous-valued sensors," *ACM Transactions on Computer Systems (TOCS),* vol. 8, no. 4, pp. 284-304, 1990.

[67] K. R¨Omer, P. Blum, and L. Meier, "Time synchronization and calibration in wireless sensor networks," in *Handbook of Sensor Networks: Algorithms and Architectures*, I. Stojmenovic, Ed., Hoboken, NJ: John Wiley & Sons, pp. 199—237, 2005.

[68] P. Chew and K. Marzullo, "Masking failures of multidimentional sensors," in *Proceedings of the 10th Symposium on Reliable Distributed Systems. IEEE*, Pisa, Italy, pp. 32-41, 1991.

[69] U. Schmid and K. Schossmaier, "How to reconcile fault-tolerant interval intersection with the Lipschitz condition," *Distributed Computing*, vol. 14, no. 2, pp. 101-111, 2001.

[70] A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer*, vol. 22, no. 6, pp. 46-57, 1989.

[71] M. Ribo and A. Pinz, "A comparison of three uncertainty calculi for building sonar-based occupancy grids," *Robotics and Autonomous Systems*, vol. 35, no. 3-4, pp. 201—209, 2001.

[72] A. Hoover and B. Oslen, “Sensor network perception for mobile robotics,” in *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 342-347, 2000.

[73] C. Wongngamnit and D. Angluin, “Robot localization in a grid,” *Information Processing Letters*, vol. 77, no. 5-6, pp. 5-6, 2001.

[74] D. Pagac, E. Nebot, and H. Durrant-Whyte, “An evidential approach to map-building for autonomous vehicles,” *IEEE Trans. Robotics Autom*, vol. 14, no. 4, pp. 623—629, 1998.

[75] Y. Zhao, R. Govindan, and D. Estrin, “Residual energy scans for monitoring wireless sensor networks,” in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'02), IEEE*, Orlando, FL, pp. 356—362, 2002.

[76] D. Koks and S. Challa, “An introduction to bayesian and dempster-shafer data fusion,” *Defence Science and Tech Org*, 2003.

[77] A. Ibrahim, *Fuzzy Logic for Embedded Systems Applications*, Newton, MA: Butterworth-Heinemann, 2003.

[78] W. Sung, and C. Hsiao, “IHPG algorithm for efficient information fusion in multi-sensor network via smoothing parameter optimization,” *Informatica*, vol. 24, no. 2, pp. 219-230, 2013.

[79] P. Castelaz, "Neural networks in defense applications," in *Proceedings of the IEEE International Conference on Neural Networks, IEEE,* San Diego, CA, pp. 473-480, 1988.

[80] W. Sung, and M. Tsai, "Multi-sensor wireless signal aggregation for environmental monitoring system via multi-bit data fusion," *Applied Mathematics&Information Sciences*, vol. 5, no. 3, pp. 589-603, 2011.

[81] Y. Zeng, J. Zhang, and J. Genderen, "Comparison and analysis of remote sensing data fusion techniques at feature and decision levels," in *ISPRS 2006 : ISPRS mid-term symposium 2006 remote sensing : from pixels to processes*, 2006.

[82] F. Castanedo, "A Review of Data Fusion Techniques," *The Scientific World Journal*. Article ID 704504. 19 pages, 2013.

[83] C. Yang, S. Bagchi, and W. Chappell, "Location tracking with directional antennas in wireless sensor networks," in *2005 IEEE MTT-S International Microwave Symposium Digest. IEEE*, Long Beach, CA, 2005.

[84] M. Ghahroudi and R. Sabzevari, "Multisensor Data Fusion Strategies for Advanced Driver Assistance Systems," in *Sensor Data Fusion; I-Tech Education and Publishing KG*, pp. 141-166, 2009.

[85] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. Jordan, and S. Sastry, "Kalman filtering with intermittent observations," *IEEE Trans. Autom. Cont*., vol. 49, no. 9, pp. 1453-1464, 2004.

[86] M. Paskin and S. Thrun, “Robotic mapping with polygonal random fields,” in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 450–458, 2005.

[87] K. Maraiya, K. Kant, and N. Gupta, “Study of Data fusion in Wireless Sensor Network,” in *Proceedings of the International Con-ference on Advanced Computing and Communication Technologies (ACCT 2011)*, Ro'htak, pp. 535-539, 2011.

[88] I. Ullah, F. Ullah, Q. Ullah, and S. Shin, “Integrated tracking and accident avoidance system for mobile robots”, *International Journal of Control, Automation and Systems*, pp.1253-1265, 2013.

[89] N. D. Phuc and N. T. Thinh, “A solution of obstacle collision avoidance for robotic fish based on fuzzy systems,” in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO*), pp. 1707–1711, Dec. 2011.

[90] J. Tian, M. Gao, and E. Lu, “Dynamic Collision Avoidance Path Planning for Mobile Robot Based on Multi-sensor Data Fusion by Support Vector Machine,” *Mechatronics and Automation, ICMA* 2007, pp. 2779-2783, 2007.

[91] (2015). *Cyberbotics.com, Webots*, accessed on Mar. 31, 2015. [Online]. Available: https://www.cyberbotics.com/overview.

[92] N. B. Hui, V. Mahendar, and D. K. Pratihar, “Time-optimal, collision-free navigation of a car-like mobile robot using neuro-fuzzy approaches,” *Fuzzy Sets and Systems*, vol. 157, no. 16, pp.2171-2204, 2006.

[93] A. Al-Mayyahi, W. Wang, and P. Birch, “Adaptive Neuro-Fuzzy Technique for Autonomous Ground Vehicle Navigation,” *Robotics* 2014, vol. 3, pp.349-370, 2014.

[94] M.L. Anjum, J. Park, W. Hwang, H. Kwon, J. Kim, C. Lee, K. Kim, and D. Cho, “Sensor data fusion using Unscented Kalman Filter for accurate localization of mobile robots,” *Control Automation and Systems (ICCAS)*, pp.947-952, 2010.

[95] E. Nada, M. Abd- Allah, M. Tantawy, and A. Ahmed, “Teleoperated Autonomous Vehicle,” *International Journal of Engineering Research & Technology (IJERT)*, Vol. 3, no. 7, 2014.

[96] Y. H. Kim, S. C. Ahn, and W. H. Kwon, “Computational complexity of general fuzzy logic control and its simplification for a loop controller,” *Fuzzy Sets Sys*, vol. 111, no. 2, pp. 215-224, 2000.

[97] S.K. Harisha, R. Kumar, P.M. Krishna and S.C. Sharma, "Fuzzy logic reasoning to control mobile robot on pre-defined strip path," *World Academy of Science Engineering and Technology*, vol. 42, 2008.

[98] M. I. Ibrahim, N. Sariff, J. Johari and N. Buniyamin, "Mobile robot obstacle avoidance in various type of static environments using fuzzy logic approach," *Electrical, Electronics and System Engineering (ICEESE), 2014 International Conference on*, Kuala Lumpur, pp. 83-88, 2014.

# PUBLICATIONS

## Journal Publications:

- Marwah Almasri, Khaled Elleithy, Abrar Alajlan, "Sensor Fusion Based Model for Collision Free Mobile Robot Navigation". ***Sensors*** 2016, *16*, 24. (Impact factor 2.24).
- Marwah Almasri, Abrar M. Alajlan, and Khaled M. Elleithy, "Trajectory Planning and Collision Avoidance Algorithm for Mobile Robotics System", in *IEEE Sensors Journal*, vol. 16, no. 12, pp. 5021-5028, June15, 2016. *(*Impact factor 1.88).
- Marwah Almasri, Khaled Elleithy, "Data Fusion in WSNs: Architecture, Taxonomy, Evaluation of Techniques, and Challenges", *International Journal of Scientific and Engineering Research(IJSER).* April, 2015.

## Conference Publications:

- Marwah Almasri and Khaled Elleithy, "Data Fusion Models in WSNs: Comparison and Analysis", In *the proceedings of the American Society for Engineering Education (ASEE ) Zone I Conference*, Bridgeport, CT, USA, April 2014.
- Abrar M. Alajlan, Marwah M. Almasri, Khaled M. Elleithy, "Multi-Sensor Based Collision Avoidance Algorithm for Mobile Robot", *IEEE Long Island Systems, Applications and Technology Conference*, March 2015. (Best Paper Award).

- Marwah Almasri, Khaled Elleithy, Abrar Alajlan, “Development of Efficient Obstacle Avoidance and Line Following Mobile Robot with the Integration of Fuzzy Logic System in Static and Dynamic Environments”, *IEEE Long Island Systems, Applications and Technology Conference*, April 2016.
- Marwah Almasri, Khaled Elleithy," Multi Sensor Fusion Based Framework for Efficient Mobile Robot Collision Avoidance and Path Following System,” *The 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2017)*, 5-9 March, 2017.

**Posters:**

- Marwah Almasri, Khaled Elleithy, and Abrar Alajlan, " Fuzzy Logic Control for Autonomous Mobile Robots in Static and Dynamic Environments,“ Poster presentation at *Faculty Research Day (FRD),* University of Bridgeport, April 1st , 2016. (3rd Place Award).
- Marwah Almasri, Khaled Elleithy, and Abrar Alajlan, “Path Planner Mobile Robot with Collision Avoidance Technique,“ Poster presentation at *Annual IEEE Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA 2016),* University of Bridgeport, Oct 14, 2016. (Honorable Mention Award).
- Abrar Alajlan, Khaled Elleithy, and Marwah Almasri " Energy-Efficient Dynamic Motion Control for Wheeled Mobile Robots Using Low Cost Resources ,“ Poster presentation at *Northeast Section ASEE Conference* at University of Rhode Island, April 28 - 30, 2016. (4th place Poster award).

(Displayed at a reception in Bridgeport City Hall hosted by Bridgeport mayor, April 7, 2016).

- Marwah Almasri, Abrar Alajlan, Khaled Elleithy,"Collision Avoidance Algorithm for Multi-sensor Mobile Robot ," Poster presentation at *Northeast Section ASEE Conference* at Northeastern University, April 30–May 2, 2015.
- Alajlan, A., Elleithy, K., Almasri., M, "Multi-Sensor Dynamic Motion Planning and Collision Avoidance for Autonomous Mobile Robot," *Annual IEEE Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA 2016) at University of Bridgeport,* Oct 14. (Honorable Mention Award).